Revisiting Secure Two-Party Computation with Rational Players

Arpita Maitra, Goutam Paul, and Asim K. Pal

Abstract—A seminal result of Cleve (STOC 1986) showed that fairness, in general, is impossible to achieve in case of two-party computation if one of them is malicious. Later, Gordon et al. (STOC 2008, JACM 2011) observed that there exist two distinct classes of functions for which fairness can be achieved. One is any function without an embedded XOR, and the other one is a particular function containing an embedded XOR. In this paper, we revisit both classes of functions in two-party computation under rational players for the first time. We identify that the protocols proposed by Gordon et al. achieve fairness in non-rational setting only. In this direction, we design two protocols, one for the millionares' problem or the greater-than function (any function without embedded XOR can be converted to this function) and the other for the particular embedded XOR function of Gordon et al., and show that with rational players, our protocols achieve fairness, correctness and strict Nash equilibrium under suitable choice of parameters in complete information game setting. The dealer is offline in both of our protocols and this is in contrast with the work of Groce et al. (Eurocrypt 2012) which shows fairness and Bayesian Nash equilibrium in two party computation with rational players for arbitrary function in an incomplete information game setting.

Index Terms—Cryptography, embedded XOR, fairness, millionaires' problem, secure computation.

1 Introduction

In a secure two-party computation, two parties or players want to compute a particular function of their inputs while preserving specific security notions under certain adversarial model. In [7], Cleve showed an impossibility result that certain functions cannot be computed with complete fairness without an honest majority. From this, the community conjectured that no function can be computed without an honest majority. However, in [4], [6] the authors showed that absolute correctness can be achieved for certain other types of functions in case of multi-party computation with one-third faulty players. The positive results of [4], [6] do not conflict with the negative result of [7] as the impossibility result is shown for a different type of functions. The solution proposed in [4], [6] consider broadcast channel model. After more than two decades, Gordon et al. [9] came out with two sets of functions for which complete fairness is possible for two-party computation in non-simultaneous channel model, even if one of the players is malicious.

One particular function of interest in [9] was the Yao's millionaires' problem [22], or more precisely, the 'greater than' function. The problem deals with two millionaires, Alice and Bob, who are interested in finding who amongst them is richer, without revealing their actual wealth to each other. Since the subsequent work [10] by Gordon et al. showed that any function over polynomial-size domains which does not contain an "embedded XOR" can be converted into the greater than function, the millionaires' problem covers all functions without embedded XOR.

In this paper, for the first time we study the fairness and correctness in millionaires' problem with rational players. Rational players are neither 'good' nor 'malicious', they are utility maximizing. Each rational party wishes to learn the output while allowing as few others as possible to learn the output. Thus, each rational party chooses abort to maximize its utility. We show that the solution of Gordon et al. for millionaires' problem in non-rational setting no longer remains fair in rational setting. We also propose a modification in the protocol with the help of a third player (explained later) so that fairness, correctness and strict Nash equilibrium can be established.

The work by Gordon et al. [9], [10] also studied the function that belongs to the class of embedded XOR. The XOR function simply checks whether the inputs chosen by two players (from a specified domain) are equal or not. They showed that under certain parameter value of a hybrid model, fairness is achieved. In this paper, we also revisit this problem with rational players and show that fairness is no longer guaranteed. We propose a modified version of the protocol by Gordon et al. in non-rational setting and prove its fairness and strict Nash equilibrium under suitable choice of the parameters.

In [3], Asharov et al. gave the full characterization of the functions which never be computed with complete fairness in two party setting when one of the parties is malicious. They actually extended the negative result of Cleve [7]. Interestingly, neither the greater than function nor the embedded XOR function belongs to these groups.

1.1 Related Works

In [2] it is shown that it is impossible to compute a function in two party setting with complete fairness if the players

- *A. Maitra and A. K. Pal are with Management Information Systems Group, Indian Institute of Management Calcutta, India. Email: {arpitam, asim}@iimcal.ac.in*
- *G. Paul is with Cryptology & Security Research Unit, R. C. Bose Centre for Cryptology & Security, Indian Statistical Institute, Kolkata. Email: goutam.paul@isical.ac.in*

are rational. However, in [12] the authors identified that the impossibility results of [2] are valid for some specific functions, specific input distribution and for specific set of utilities. They [12] came out with an algorithm for arbitrary function which can be computed with complete fairness in two party setting considering rational players. However, their protocol is an *incomplete information game* and thus achieves *Bayesian Nash equilibrium*. Contrary to this, our proposed protocols are a *complete information game* and achieves *strict Nash equilibrium*. Moreover, in [12], the following is mentioned.

> "Before continuing, it is helpful to introduce two modifications to the protocol that can only increase P_0's utility. First, in each iteration i we tell P_0 whether $i^* < i$. (One can easily see that P_0 cannot increase its utility by aborting when $i^* < i$, and so the interesting case to analyze is whether P_0 can improve its utility by aborting when $i^* \geq i$.) Second, if P_0 ever aborts the protocol in some iteration i with $i^* \geq i$, then we tell P_0 whether $i^* = i$ before P_0 generates its output. (P_0 is not, however, allowed to change its decision to abort.)"

It is not clear how "we" can let the player know whether $i^* < i$ or $i^* \geq i$. Note that in [12], the symbol i stands for any iteration in which player P_0 (in our case it is P_1) aborts the protocol and symbol i^* stands for the revelation round (in our case it is r). One may interpret this "we" as a dealer who has to remain online throughout the game. This is not very practical, as in each iteration the dealer has to interact with the players and has to ask them whether they will choose to abort. Another restriction in their scheme is that the deviating player can not escape from its decision, knowing that the round it has chosen to abort is less than or equal to the revelation round. Exploiting the idea of an intermediate player who is a different entity from the dealer, we are able to make the dealer off-line in the millionares' problem. Note that our protocol for the embedded XOR requires neither an online dealer, nor an intermediate player.

One may think that the use of a third player is no different than the use of a dealer. But the fact is that our third player is less restrictive in comparison with the dealer. In the literature, the dealer is always assumed to be honest and this is a strong assumption. On the other hand, the third player in our model is assumed to be rational in nature and this is a more pragmatic assumption. Moreover, the dealer is a special distinct entity from the players, whereas the role of our intermediate third player can be adopted by any rational player who is not interested in the outcome of the game, rather his only objective is to earn some revenue at the end of the game. The analogy can be of a referee conducting a match. A referee remains true to the game and only deviates for a huge (monitory) gain at the risk of losing his reputation or job. A third player can also be referred as a 'service provider' entity, e.g. a cloud or an outsourcing agent. For simplicity, we assume that this third player is fail-stop in nature.

Making the dealer offline in the rational secret sharing was once a challenging task and it was overcome by introducing an indicator bit [14] or a signal [8] to make the player aware of the fact that they have passed the revelation round. To the best of our knowledge, there is no protocol in two party secure computation with rational players where the dealer is made offline. We believe that introduction of a third player is not a weakness of our protocol for the millionares' problem, rather it may be considered as the first effort to make the dealer offline in secure two party protocol with rational players.

In case of embedded XOR problem, one may easily notice that when a player chooses x_3 as his input, he knows with certainty that his output will be 1. So he should have no incentive to play the game anymore. According to [12], if a party aborts the game, the other party outputs 0 or 1 depending on his input. Thus, there is a non-negligible probability that the protocol of [12] does not achieve fairness. On the other hand, our proposed protocol for the embedded XOR problem is free from this problem. We summarize all the positive and negative results in the context of multiparty computation in Table 1.

1.2 Contributions

We list our key contributions one by one.

1) We revisit fairness in two prominent Secure Two-Party Computation problems, namely, Yao's millionaires' problem [22] and the Embedded XOR problems [9], [10], for the first time with rational players.
2) We show that the protocol of Gordon et al. [9], [10] for solving the millionaires' problem which was meant for the non-rational setting does not automatically extend to the rational setting, in the sense that the fairness breaks down when the players are rational (Theorem 1).
3) We propose a variant of this protocol and show that fairness can be regained when players are rational (Theorem 4). We also establish correctness (Theorem 3) and strict Nash equilibrium (Theorem 5) for the new protocol.
4) We get away with the online dealer by introducing a rational third party for solving the millionaires' problem. This helps us to establish the fairness of our protocol.
5) We show that the problem in the embedded XOR category [9], [10] also no longer remains fair with rational players (Theorem 6).
6) We propose a variant of the protocol in [9], [10] for embedded XOR and show that fairness can be guaranteed under certain assumptions (Theorem 7). We also establish strict Nash equilibrium for our protocol (Theorem 8).
7) For both the problems, we discuss the issues with unequal vs. equal domain sizes.

2 Preliminaries

In this section we try to explain what is meant by functionality, two party computation, ideal and real world model, Byzantine and fail-stop adversary. We also define utilities and fairness in rational setting which is used in this work.

TABLE 1
Possibility and Impossibility Results in the Context of Multiparty Computation

Authors	Year / Venue	Model	Possibility Results	Maximum Number of faulty players	Security Notions
Cleve [7]	1986 STOC	Non-rational (Malicious)	Impossibility result : for certain functions	One out of two	Complete fairness
Ben -Or et al. Chaum et al. [4], [6]	1988 STOC	Non-rational (Malicious)	Positive result: for certain other functions	$\frac{n}{3}$ out of n	Complete correctness
Gordon et al. [9]	2008 STOC	Non-rational (Malicious)	Positive results for two distinct sets of functions	One out of two	Complete fairness
Asharov et al. [2]	2011 Eurocrypt	Rational, Incomplete information game, Incentive incompatible	Impossibility result: for a specific function	One out of two	Complete fairness
Groce et al. [12]	2012 Eurocrypt	Rational, Incomplete information game, Incentive compatible	Positive result: any function	One out of two	Complete fairness, Bayesian Nash equilibrium
Asharov et al. [3]	2013 TCC	Non-rational (Malicious)	Impossibility result: full characterization of the functions (extension of Cleve)	One out of two	Complete fairness
Proposed	2016	Rational, Complete information game, Incentive compatible	Positive results: two distinct types of functions considered by Gordon et al.	One out of two	Fairness, Correctness (for the greater than function), Strict Nash equilibrium

2.1 Functionality

In classical domain and in two party setting, a functionality $\mathcal{F} = \{f_\lambda\}_{\lambda \in \mathbb{N}}$ is a sequence of randomized processes, where λ is the security parameter and f_λ maps pairs of inputs to pairs of outputs (one for each party). Explicitly, we can write $f_\lambda = (f_\lambda^1, f_\lambda^2)$, where f_λ^1 (resp., f_λ^2) represents the output of the first party, say P_1 (resp., output of the second party, say P_2). The domain of f_λ is $X_\lambda \times Y_\lambda$, where X_λ (resp. Y_λ) denotes the possible inputs of the first (resp. second) party. If $|X_\lambda|$ and $|Y_\lambda|$ are polynomial in λ, then we say that $\mathcal{F}$ is defined over polynomial size domains. If each f_λ is deterministic we say that each f_λ as well as the collection $\mathcal{F}$ is a function [10].

2.2 Two Party Computation

In classical domain the two party computation of a functionality $\mathcal{F} = \{f_\lambda^1, f_\lambda^2\}$ is defined as follows: If party P_1 is holding 1^λ and a input $x \in X_\lambda$ and party P_2 is holding 1^λ and a input $y \in Y_\lambda$, then the joint distribution of the outputs of the parties is statistically close to $(f_\lambda^1(x, y), f_\lambda^2(x, y))$ [10].

2.3 Ideal vs. Real world model

In *ideal world model* we assume that there is an incorruptible trusted third party who computes the function on behalf of P_1 and P_2. P_1 and P_2 send their inputs to the TTP who computes the functionality and returns the value to each party. On the other hand, in *real world model* there is no trusted party to compute the functionality. In *real model* a protocol is executed to compute the functionality.

In the same line of [9], [10], we here assume a *hybrid world model*, where there is a trusted party who just computes the function and distributes the shares of the function's output as in the ideal world. This is the counterpart of the case in secret sharing [16] in the non-rational setting.The players construct the output by exchanging their shares. In our model, we call this TTP as *dealer*.

2.4 Rational secret sharing

Secure Multiparty Computation (SMC) is the generalization of the classical rational secret sharing [12]. In this subsection, we briefly describe what is meant by classical rational secret sharing.

Rational secret sharing proceeds in two phases: 1) share generation and distribution and 2) secret reconstruction. Dealer generates the shares of the secret and distributes among the players in the 1st phase. One important difference to be noted here that in the rational setting there can be multiple shares including fake shares in contrast to just one share in the non-rational setting. The dealer in a classical rational secret sharing (RSS) protocol is honest and can be online or offline. An online dealer remains available throughout the secret reconstruction protocol i.e in the 2nd phase, whereas an offline dealer is unavailable after distributing the shares of the secret (i.e unavailable after 1st phase). Note that an online dealer is not very practical as he repeatedly interacts with the players and such a dealer can directly provide the secret to the players. In 2008, Kol and Naor [14] discussed rational secret sharing in the non-simultaneous channel model and in the presence of an offline dealer, in an information theoretic setting. Almost all the subsequent works [1], [21], [17], [8] on rational secret sharing assumed the dealer to be offline.

Share generation and distribution: If the dealer is online, then at the beginning of each round, he distributes to each player P_w the share of the actual secret with probability γ or that of a fake secret with probability $(1-\gamma)$. The value of γ is kept secret from the parties and is dependent on the utility values of the parties [13], [11]. An offline dealer distributes to each party P_w a list of shares, one of which is that of the actual secret s and the remaining of fake secrets [14], [8], [17]. The position r of this actual share in the lists is not revealed to the players and is chosen according to a geometric distribution $\mathcal{G}(\gamma)$, where the parameter γ in turn depends on the utility values of players. The dealer generates shares using Shamir's secret sharing scheme [16].

Secret Reconstruction: In the lth round of communication, each player P_w (either simultaneously or non-simultaneously) broadcasts or sends individually to each of the other players (in presence of synchronous, point-to-point channels) the share s_{wl} corresponding to that round. The shares are signed by the dealer. Hence, no player can give out false shares undetected and the only possible action of a player in a round is to either 1) send the message or 2) remain silent. The round in which the shares of the actual secret are revealed and hence the secret is reconstructed is called revelation or definitive round. When the dealer is offline, players are made aware that they have crossed the revelation round by the reconstruction or exchange of an indicator (a bit in [14], a signal in [8]). For simultaneous channel model, parties can identify a revelation round as soon as it occurs. However, for non-simultaneous channels, the indication is delayed till the subsequent round to avoid rushing strategy. In this case, the indicator cannot be reconstructed or interpreted by all the players. The player who communicates last during the reconstruction of the indicator is the first and only one to know that the last round was the revelation round. Once he comes to know this, he has no incentive to send his share of the indicator to the other players for reconstruction. Instead, he simply quits. The fact that this player quits signals to the other players that the secret has been reconstructed.

2.5 Computation of a functionality in Rational setting

We define a *mechanism for computing a functionality with rational adversary* to be a pair $(\Gamma, \overrightarrow{\sigma})$, where Γ is the game (i.e., specification of allowable actions) and $\overrightarrow{\sigma}$=$(\sigma_1, \ldots, \sigma_n)$ denotes the suggested strategies followed by n number of players. We use the notations $\overrightarrow{\sigma}_{-w}$ and $(\sigma'_w, \overrightarrow{\sigma}_{-w})$ respectively for $(\sigma_1, \ldots, \sigma_{w-1}, \sigma_{w+1}, \ldots, \sigma_n)$ and $(\sigma_1, \ldots, \sigma_{w-1}, \sigma'_w, \sigma_{w+1}, \ldots, \sigma_n)$. Here, σ'_w stands for deviated strategy. The outcome of the game is denoted by $\overrightarrow{o}(\Gamma, \overrightarrow{\sigma})$=$(o_1, \ldots, o_n)$. The set of possible outcomes with respect to a party P_w is as follows. 1) P_w correctly computes f, while others do not; 2) everybody correctly computes f; 3) nobody computes f; 4) others computes f correctly, while P_w does not and 5) others believe in a wrong functional value, while P_w does not.

The output that no function is computed is denoted by $\perp$ (i.e., *null* as in [9]) and output of wrong computation is denoted by $\neg$.

In classical domain, the adversary that controls a player may be computationally bounded. Here, we assume the adversary has probabilistic polynomial time complexity.

2.6 Utilities and Preferences

The utility function u_w of each party P_w is defined over the set of possible outcomes of the game. The outcomes and corresponding utilities for two parties are described in Table 2. We here assume Bernoulli utility function.

TABLE 2
Outcomes and Utilities for $(2,2)$ rational function reconstruction

P_1's outcome (o_1)	P_2's outcome (o_2)	P_1's Utility $U_1(o_1, o_2)$	P_2's Utility $U_2(o_1, o_2)$
o_1=f	o_2=f	U_1^{TT}	U_2^{TT}
o_1=$\perp$	o_2=$\perp$	U_1^{NN}	U_2^{NN}
o_1=f	o_2=$\perp$	U_1^{TN}	U_2^{NT}
o_1=$\perp$	o_2=f	U_1^{NT}	U_2^{TN}
o_1=$\perp$	o_2=$\neg$	U_1^{NF}	U_2^{FN}
o_1=$\neg$	o_2=$\perp$	U_1^{FN}	U_2^{NF}

Players have their preferences based on the different possible outcomes. In this work, a rational player w is assumed to have the following preference:

$$\mathcal{R}_1 : U_w^{TN} > U_w^{TT} > U_w^{NN} > U_w^{NT}.$$

Some players may have the additional preference $U_w^{NF} \geq U_w^{TT}$, whereas the rest have $U_w^{NF} < U_w^{TT}$.

2.7 Fairness

In non-rational setting, the security of a protocol is analyzed [9], [10], [15] by comparing what an adversary can do in a *real* protocol execution to what it can do in an *ideal* scenario that is secure by definition. This is formalized by considering an ideal computation involving an incorruptible trusted party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it were involved in the above-described ideal computation.

A rational player, being selfish, desires an unfair outcome, i.e., computing the function alone. Therefore, the basic aim of rational computation has been to achieve fairness. According to Von Neumann and Morgenstern *expected utility theorem* [20], under natural assumptions, the individual would prefer one prospect $\mathcal{O}_1$ over another prospect $\mathcal{O}_2$ if and only if $E[U(\mathcal{O}_1)] \geq E[U(\mathcal{O}_2)]$. The work [1] implicitly uses the expected utility theorem to derive its results. We also use the same approach and accordingly redefine fairness as follows.

Definition 1. *(Fairness) A function reconstruction mechanism* $(\Gamma, \overrightarrow{\sigma})$ *with rational players is said to be completely fair if for a party* P_w*,* $(w \in \{1,2\})$*, who is corrupted by a probabilistic polynomial time adversary, the following holds:*

$$U_w^{TT} \geq E[U_w(\mathcal{O}_l)],$$

where $\mathcal{O}_l = \{o_w^1, \ldots, o_w^{n'}; p_1, \ldots, p_{n'}\}$ *is a prospect when the player deviates from the suggested strategy* (σ_w)*.* n' *is the number of possible outcomes.*

2.8 Correctness of the Output

Secure two party computation in *hybrid model* with rational players is the generalization of rational secret sharing [11], [13], [14]. In rational secret sharing, the dealer is not a part of the reconstruction mechanism and therefore the output is well-defined. Whereas, in secure two party computation, the parties may not be committed to their inputs, and therefore the output is not uniquely defined. We here assume that the players have negligible probability to send arbitrary inputs. This is quite justified in the sense that the players are rational in nature, i.e., each wants to compute the function itself and does not want anyone else to compute the function. If a player sends wrong inputs, then it itself cannot learn the correct output. Thus, the players have no motivation to send wrong inputs. Rather they try to maximize their utility [12].

2.9 Complete Information Game and Nash Equilibrium

In a complete information game each player knows the other player's payoff function and the rule of the game. It is defined as $\Gamma = (A_w, u_w)$, where A_w is the set of allowable actions of each player w and u_w is the utility function of the player w [1].

A suggested strategy $\overrightarrow{\sigma}$ of a mechanism $(\Gamma, \overrightarrow{\sigma})$ is said to be in Nash equilibrium when there is no incentive for a player P_w to deviate from the suggested strategy, given that everyone else is following this strategy. There are several variants of Nash equilibrium in the literature. In our context, we focus on strict Nash equilibrium.

Definition 2. *(Strict Nash equilibrium) The suggested strategy $\overrightarrow{\sigma}$ in the mechanism $(\Gamma, \overrightarrow{\sigma})$ is a strict Nash equilibrium if for every P_w and for any strategy σ'_w, we have $u_w(\sigma'_w, \overrightarrow{\sigma}_{-w}) < u_w(\overrightarrow{\sigma})$.*

In computational Nash equilibrium [1], the players will not change their strategy if their gain is negligible [5]. Since we show strict Nash for our protocols and strict Nash implies computational Nash, henceforth we do not talk about computational Nash anymore.

2.10 Fail-stop and Byzantine Adversarial model

In the fail-stop setting, each party follows the protocol as directed except that it may choose to abort at any time [12] and a party is assumed not to change its input when running the protocol. On the other hand, in Byzantine setting, a deviating party may behave arbitrarily. It may change the inputs or may choose to abort. Since Byzantine adversary covers all the characteristics of a fail-stop adversary, it is very natural to consider only Byzantine setting. If a protocol is secure against a Byzantine adversary, it must be secure against a fail-stop adversary. Hence, throughout the paper we analyze the security issues against Byzantine adversary only.

2.11 Our Assumptions

Here we list the assumptions that we consider in this work.

1) The channel permits non-simultaneous mode of communication.
2) Players are computationally bounded to probabilistic polynomial time complexity.
3) We consider complete information game unlike [12].
4) The same utility relationship holds as considered by Groce et al. [12]. In addition, the preference of $U_w^{NF} \geq U_w^{TT}$ [1] is considered.
5) The adversary may be fail-stop as well as Byzantine.

3 SMC for Functions Excluding Embedded XOR with Rational Players in Equal Domain Size

In this section, we first describe the solution of millionaires' problem or, more precisely, the computation of the greater than function, proposed by Gordon et al. [9], [10]. We, then, will show how fairness condition is affected in the presence of the rational players having the preferences $\mathcal{R}_1$ (refer to subsection 2.6). Let us denote two players by P_1 and P_2. Suppose P_1 has the secret i and P_2 has the secret j, $1 \leq i \leq M$, $1 \leq j \leq M$, where $M = M(\lambda)$ is the size of the domain of each input [9]. The trusted third party in *hybrid model* gives an ordered list $X = \{x_1, x_2, \ldots, x_M\}$ to P_1 and another ordered list $Y = \{y_1, y_2, \ldots, y_M\}$ to P_2. We call this TTP as dealer. Then P_1 sends x_i to the dealer and P_2 sends y_j to the dealer. Let f be a deterministic function which maps $X \times Y \rightarrow \{0,1\} \times \{0,1\}$. The function $f(x_i, y_j)$ can be defined as a pair of outputs, i.e., $f(x_i, y_j) = (f_1(x_i, y_j), f_2(x_i, y_j))$, where $f_1(x_i, y_j)$ is the output of the first party and $f_2(x_i, y_j)$ is the output of the second party. For millionaires' problem, the function is defined as follows [9], [10]. For $w = 1, 2$,

$$f_w(x_i, y_j) = \begin{cases} 1 & \text{if } i > j; \\ 0 & \text{if } i \leq j. \end{cases} \quad (1)$$

The protocol proceeds in a series of M iterations. The dealer creates two sequences $\{a_l\}$ and $\{b_l\}$, $l = 1, 2, \ldots, M$, as follows.

$$a_i = b_j = f_1(x_i, y_j) = f_2(x_i, y_j).$$

For $l \neq i$, $a_l = \perp$ and for $l \neq j$, $b_l = \perp$.

Next, the dealer splits the secret a_l into the shares a_l^1 and a_l^2, and the secret b_l into the shares b_l^1 and b_l^2, so that $a_l = a_l^1 \oplus a_l^2$ and $b_l = b_l^1 \oplus b_l^2$, and gives the shares $\{(a_l^1, b_l^1)\}$ to P_1 and the shares $\{(a_l^2, b_l^2)\}$ to P_2. In each round l, P_2 sends a_l^2 to P_1, who, in turn sends b_l^1 to P_2. P_1 learns the output value $f_1(x_i, y_j)$ in iteration i, and P_2 learns the output in iteration j. As we require three elements, 0, 1 and $\perp$, we define 0 by 00, 1 by 11 and $\perp$ by 01. The algorithm for the functionality share generation in fail-stop setting is revisited in Algorithm 1. Here we assume that the dealer who will distribute the shares is honest and can compute the function described in Equation (1). The protocol for computing f is described in Algorithm 2.

The algorithms in the Byzantine setting are the same as those in the fail-stop setting except some additional steps. In Byzantine setting, the shares are signed by the dealer. Along with the shares of the function, the dealer also distributes some secret keys $k_a, k_b \leftarrow Gen(1^\lambda)$, where λ is the security parameter. For $1 \leq l \leq M$, let $t_l^a = Mac_{k_a}(l \parallel a_l^2)$ and $t_l^b = Mac_{k_b}(l \parallel b_l^1)$. P_1 receives $a_1^1, a_2^1, \ldots, a_M^1$ and $(b_1^1, t_1^b), (b_2^1, t_2^b), \ldots, (b_M^1, t_M^b)$ and MAC key k_a. Similarly P_2 is given $(a_1^2, t_1^a), (a_2^2, t_2^a), \ldots, (a_M^2, t_M^a)$ and $b_1^2, b_2^2, \ldots, b_M^2$ and MAC key k_b. After receiving the share in the round l

Inputs:
x_i from P_1 and y_j from P_2. If one of the received input is not in the correct domain, then both the parties are given $\perp$.
Computation:
The dealer does the following:
Sets $a_i = b_j = f_1(x_i, y_j) = f_2(x_i, y_j)$.
For $l \in \{1, \ldots, M\}, l \neq i$, sets $a_l = \perp$.
For $l \in \{1, \ldots, M\}, l \neq j$, sets $b_l = \perp$.
For $l \in \{1, \ldots, M\}$, chooses a_l^1 randomly from $\{0,1\}^2$, and sets $a_l^2 = a_l^1 \oplus a_l$.
For $l \in \{1, \ldots, M\}$, chooses b_l^1 randomly from $\{0,1\}^2$, and sets $b_l^2 = b_l^1 \oplus b_l$.
Output:
The dealer prepares a list $list_w$ of shares for each party P_w, where $w \in \{1, 2\}$ such that
P_1 receives the values of $a_1^1, a_2^1, \ldots, a_M^1$ and $b_1^1, b_2^1, \ldots, b_M^1$.
P_2 receives the values of $a_1^2, a_2^2, \ldots, a_M^2$ and $b_1^2, b_2^2, \ldots, b_M^2$.

Algorithm 1: ShareGen

Inputs:
P_1 obtains $a_1^1, a_2^1, \ldots, a_M^1$ and $b_1^1, b_2^1, \ldots, b_M^1$.
P_2 obtains $a_1^2, a_2^2, \ldots, a_M^2$ and $b_1^2, b_2^2, \ldots, b_M^2$.
Computation:
There are M number of iterations. In each iteration $l \in \{1, 2, \ldots, M\}$ do:
P_2 sends a_l^2 to P_1 and P_1 computes $a_l = a_l^1 \oplus a_l^2$.
P_1 sends b_l^1 to P_2 and P_2 computes $b_l = b_l^1 \oplus b_l^2$.
Output:
If P_2 aborts in round l, i.e., does not send its share at that round and $l \leq i$, P_1 outputs 1. If $l > i$, P_1 has already determined the output in some earlier iteration. Thus it outputs that value.
If P_1 aborts in round l, i.e., does not send its share at that round and $l \leq j$, P_2 outputs 0. If $l > j$, P_2 has already determined the output in some earlier iteration. Thus it outputs that value.

Algorithm 2: $\mathbf{\Pi^{CMP}}$

from P_2, P_1 verifies by the algorithm $Vrfy_{k_a}(l \parallel a_l^2, t_l^a)$. If $Vrfy_{k_a}(l \parallel a_l^2, t_l^a) = 0$, P_1 halts. Similarly, after receiving the share in the round l from P_1, P_2 verifies by the algorithm $Vrfy_{k_b}(l \parallel b_l^1, t_l^b)$. If $Vrfy_{k_b}(l \parallel b_l^1, t_l^b) = 0$, P_2 halts. Otherwise both continues the protocol $\mathbf{\Pi^{CMP}}$ which outputs $a_i(b_j)$ for $P_1(P_2)$.

Exploiting the MAC signature, we can resist the players to send a false share.

3.1 $\mathbf{\Pi^{CMP}}$ is not fair when players are rational

In this section, we revisit the fairness issue in the millionaires' problem [9] considering the rational players. We also assume that the players, P_1 and P_2 have the preferences $\mathcal{R}_1$ (refer to subsection 2.6). Either of the players also has $U_w^{NF} \geq U_w^{TT}$. We observe that Gordon et al.'s protocol $\mathbf{\Pi^{CMP}}$ [9], [10], that was designed for non-rational setting, is no longer fair in the rational setting.

Theorem 1. *Provided $\mathcal{R}_1$ and $U_w^{NF} \geq U_w^{TT}$ for some player P_w, the protocol $\mathbf{\Pi^{CMP}}$ is not completely fair in rational setting.*

Proof. Suppose P_1 aborts before giving its share in round l, where $1 \leq l \leq M$. Now, if $i \leq j$, we list all possible mutually exclusive and exhaustive outcomes as follows:

1) When $1 \leq l < i$, P_2 outputs 0 and correctly concludes that $i \leq j$, but P_1 outputs $\perp$.
2) When $i \leq l \leq M$, P_1 obtains the function and both correctly conclude that $i \leq j$.

In this case, the utility of P_1 is given by

$$U_1^{\leq} = \begin{cases} U_1^{NT} & \text{if } 1 \leq l < i; \\ U_1^{TT} & \text{if } i \leq l \leq M; \end{cases} \tag{2}$$

If $i > j$, all possible mutually exclusive and exhaustive outcomes are:

1) When $1 \leq l \leq j$, P_2 outputs 0 and wrongly concludes that $i \leq j$, but P_1 outputs $\perp$.
2) When $j < l < i$, P_1 outputs $\perp$, but P_2 correctly concludes that $i > j$.
3) When $i \leq l \leq M$, both computes the function and both correctly conclude that $i > j$.

Thus, the corresponding utility for this event is given by

$$U_1^{>} = \begin{cases} U_1^{NF} & \text{if } 1 \leq l \leq j; \\ U_1^{NT} & \text{if } j < l < i; \\ U_1^{TT} & \text{if } i \leq l \leq M; \end{cases} \tag{3}$$

Since i is known to P_1, the expected utility of P_1 is given by

$$E[U_1] = \Pr(i \leq j) \cdot E[U_1^{\leq}] + \Pr(i > j) \cdot E[U_1^{>}], \tag{4}$$

where $\Pr(i \leq j) = \frac{M-i+1}{M}$ and $\Pr(i > j) = \frac{i-1}{M}$. Here, we assume that i and j are uniformly distributed over their domains. Plugging in the values from Equation (2) and (3) into Equation (4), we get for $1 \leq l < i$, $E[U_1] = \left(\frac{M-i+1}{M}\right) U_1^{NT} + \left(\frac{i-1}{M}\right)\left(\left(\frac{l-1}{i-1}\right) U_1^{NT} + \left(\frac{i-l}{i-1}\right) U_1^{NF}\right)$, and for $i \leq l \leq M$, it is equal to $\left(\frac{M-i+1}{M}\right) U_1^{TT} + \left(\frac{i-1}{M}\right) U_1^{TT}$.

In other words,

$$E[U_1] = \begin{cases} \left(\frac{M-i+l}{M}\right) U_1^{NT} + \left(\frac{i-l}{M}\right) U_1^{NF} & \text{if } 1 \leq l < i; \\ U_1^{TT} & \text{if } i \leq l \leq M. \end{cases}$$

Note that in the first case, i.e., for $1 \leq l < i$, the second term corresponding to $i > j$ involves two sub cases, namely, $1 \leq j < l < i$ and $l \leq j < i$.

Observe that when $i \leq l \leq M$, P_1 has already obtained the secret, but by aborting it cannot increase its utility beyond U_1^{TT}.

However, when $l < i$, we may have $E[U_1] > U_1^{TT}$, depending on the value of U_1^{NF}.Thus, dependence on U_1^{NF} prevents the protocol to achieve fairness in this case. In other words, we can say that when a party aborts before it obtains the output, the only reason would be if he is significantly more interested in cheating the other party rather than him not getting it.

The analysis for P_2 is similar, except that we have the role of i and j interchanged. $\square$

3.2 How to make $\Pi^{\mathbf{CMP}}$ fair when players are rational

In this section, we propose a variant of the protocol by Gordon et al. [9], [10]. In the earlier section, we have observed that $\mathbf{\Pi^{CMP}}$ suffers from early abort. Exploiting the idea of an intermediate player P_3, we are able to make the dealer off-line. We have already discussed the difference between online dealer and the intermediate player in detail in Section 1.1. The key point is that P_3 need not to be assumed to be honest, it is just another rational player, like P_1 or P_2.

Our protocol is described in Algorithm 3 and Algorithm 4. Though our protocol initially addresses towards the millionaires' problem, it is applicable for any function which does not have any embedded XOR [10]. Our protocol is U^{NF} (refer to subsection 2.6) independent and hence correct [1]. We also prove fairness and strict Nash equilibrium for our protocol.

Inputs:
x_i from P_1 and y_j from P_2. If one of the received input is not in the correct domain, then both the parties are given $\perp$.
Computation:
The dealer does the following:
Chooses an intermediate player P_3.
Chooses r according to a geometric distribution $\mathcal{G}(\gamma)$ with parameter γ and sets r as the revelation round, i.e., the round in which the value of f is either $(0,0)$ or $(1,1)$ (refer to subsection 2.4).
Chooses d according to the geometrical distribution $\mathcal{G}(\gamma)$ and sets the total number of iterations as $m = r + d$.
The dealer does the following:
Sets $a_r = b_r = f_1(x_i, y_j) = f_2(x_i, y_j)$.
For $l \in \{1, \ldots, M\}$, $l \neq r$, sets $a_l = b_l = \perp$.
For $l \in \{1, \ldots, M\}$, chooses a_l^1 and a_l^2 randomly from $\{0,1\}^2$, and sets $a_l^3 = a_l^1 \oplus a_l^2 \oplus a_l$.
For $l \in \{1, \ldots, M\}$, chooses b_l^1 and b_l^2 randomly from $\{0,1\}^2$, and sets $b_l^3 = b_l^1 \oplus b_l^2 \oplus b_l$.
For $l \in \{1, \ldots, M\}$, chooses c_l^2 randomly from $\{0,1\}^2$, and sets $c_l^3 = c_l^2 \oplus a_l^3$ and $c_l^1 = c_l^3 \oplus b_l^3$.
Output:
The dealer prepares a list $list_w$ of shares for each party P_w, where $w \in \{1,2,3\}$ such that
P_1 receives the values of $a_1^1, a_2^1, \ldots, a_m^1$, $b_1^1, b_2^1, \ldots, b_m^1$ and $c_1^1, c_2^1, \ldots c_m^1$.
P_2 receives the values of $a_1^2, a_2^2, \ldots, a_m^2$, $b_1^2, b_2^2, \ldots, b_m^2$ and $c_1^2, c_2^2, \ldots c_m^2$.
P_3 receives the values of $c_1^3, c_2^3, \ldots, c_m^3$.

Algorithm 3: ShareGen for $\mathbf{\Pi_{fair}^{CMP}}$

Inputs:
P_1 obtains $a_1^1, a_2^1, \ldots, a_m^1$, $b_1^1, b_2^1, \ldots, b_m^1$ and $c_1^1, c_2^1, \ldots, c_m^1$.
P_2 obtains $a_1^2, a_2^2, \ldots, a_m^2$, $b_1^2, b_2^2, \ldots, b_m^2$ and $c_1^2, c_2^2, \ldots, c_m^2$.
P_3 obtains $c_1^3, c_2^3, \ldots, c_m^3$.
Computation:
There are m number of iterations. In each iteration $l \in \{1, 2, \ldots, m\}$ do the following.
P_2 sends a_l^2 to P_1 and P_1 sends b_l^1 to P_2.
After receiving the share fromP_2, P_1 sends c_l^1 to P_3, else halts.
After receiving the share fromP_1, P_2 sends c_l^2 to P_3, else halts.
P_3 computes the values of a_l^3 and b_l^3 and sends a_l^3 to P_1 and then b_l^3 to P_2.
Output:
If P_2 aborts in round l, i.e., does not send its share at that round and $l \leq r$, P_1 outputs $\perp$. If $l > r$, P_1 has already determined the output in some earlier iteration. Thus it outputs that value.
If P_1 aborts in round l, i.e., does not send its share at that round and $l \leq r$, P_2 outputs $\perp$. If $l > r$, P_2 has already determined the output in some earlier iteration. Thus it outputs that value.
If P_1 or P_2 does not send its share to P_3, P_3 outputs $\perp$ to the both of the players.
If P_3 does not send its computed share to any one of the party P_w, $w \in \{1,2\}$, in a round l, P_w chooses to abort from the very next round and the protocol will be terminated.

Algorithm 4: $\mathbf{\Pi_{fair}^{CMP}}$

We assume that P_3 has a positive threshold value of revenue. We denote this threshold value by ϵ. If any player offers him a revenue $\delta \geq \epsilon$, P_3 will help the player to get the output alone. Otherwise he will play the game according to the suggested strategy. This threshold value may depend on the reputation value of P_3. We take the idea of reputation from [19]. Thus, P_3 has following two options:

- **Option** 1: He can follow the protocol, i.e., he can send the shares to both the parties.
- **Option** 2: If any one of the players gives him a revenue $\delta \geq \epsilon$, he will send the share only to that player and help him obtain the output alone.

As P_3 is rational and hence utility maximizer, he first checks whether any one of the players indeed gives him the revenue. Without loss of generality, we assume that P_1 gives him the revenue $\delta \geq \epsilon$. In this case, P_3 will send the share only to P_1 but not to P_2. The following result shows that P_1 should have no incentive to give the revenue to P_3 in the motivation to get the output by himself only provided certain conditions hold.

Theorem 2. *Provided $\epsilon > 0$, $\epsilon \leq \delta \leq U_w^{TN} - U_w^{TT}$, $0 < \gamma < 1$ and $\gamma U_w^{TN} + (1-\gamma) U_w^{NN} < U_w^{TT}$ for all $w \in \{1,2\}$, P_3 always plays the game according to the suggested strategy.*

Proof. According to the protocol, to obtain the secret alone with the help of P_3, P_1 has to guess correctly the revelation round. Otherwise, the protocol will be terminated from the very next round and none of the players get any information about the output. P_1 will not be interested to give the money to P_3 after the revelation round as he has already got the output at the revelation round. Conditioned on the event that $r \geq l$, suppose P_1 guesses the l-th round to be the revelation round and gives P_3 the money for that round so that P_3 will not send the corresponding share to P_2 for that round. If the guess is correct, i.e., $r = l$, the probability of which is γ, its utility is $(U_1^{TN} - \delta)$. Otherwise, its utility is $(U_1^{NN} - \delta)$, as in this case P_2 will abort from the next round. So the expected utility of P_1 is given by

$$\begin{aligned} & \gamma(U_1^{TN} - \delta) + (1-\gamma)(U_1^{NN} - \delta) \\ = \; & \gamma U_1^{TN} + (1-\gamma)U_1^{NN} - \delta < U_1^{TT} - \delta < U_1^{TT}. \end{aligned}$$

The last inequality follows from our assumptions that δ is positive and $\gamma U_1^{TN} + (1-\gamma)U_1^{NN} < U_1^{TT}$. Thus P_1 has no incentive to offer money to the intermediate player P_3 in the motivation to get the output alone. Similar analysis can be done for P_2.

Thus, P_3 has no option but to play the game according to the suggested strategy. □

In our mechanism, there are three players, namely P_1, P_2 and P_3. For the condition of achieving correctness and fairness, we have to assume that when one of the players deviates, others are sticking to the protocol. From the above analysis we have seen that P_3 has no incentive to deviate from the protocol. Thus, we have to consider the following two cases.

1) P_1 deviates (P_2 follows the protocol).
2) P_2 deviates (P_1 follows the protocol).

In fail-stop setting, the deviation of P_1 and P_2 is considered as early abort whereas in Byzantine setting the players behave arbitrarily. That means they can abort early as well as can send the arbitrary inputs or can swap the inputs.

We analyze the security notions such as correctness and fairness considering all the above issues. The following theorems show that our proposed mechanism is correct and fair.

In Byzantine setting, the shares given to the players are signed by the dealer so that no player can send a false share to the other player. The signing procedure discussed in Section 3 remains similar in our protocol expect M is replaced by m and with some additional steps.

- For $1 \leq l \leq m$, P_1 is given $(c_l^1, t_l^{c_1})$, where $t_l^{c_1} = Mac_{k_{c_1}}(l \parallel c_l^1)$.
- For $1 \leq l \leq m$, P_2 is given $(c_l^2, t_l^{c_2})$, where $t_l^{c_2} = Mac_{k_{c_2}}(l \parallel c_l^2)$.
- P_3 is given MAC key k_{c_1} and MAC key k_{c_2} so that for $1 \leq l \leq m$, it can verify the shares by algorithm $Vrfy_{k_{c_1}}(l \parallel c_l^1, t_l^{c_1})$ for P_1 and $Vrfy_{k_{c_2}}(l \parallel c_l^2, t_l^{c_2})$ for P_2. If $Vrfy_{k_{c_w}}(l \parallel c_l^w, t_l^{c_w}) = 0$, P_3 halts, else continues, where $w \in \{1, 2\}$.

There is no need to sign the shares given to P_3, as P_3 is fail-stop by nature. We assume P_3 as a fail-stop player for simplicity. One may consider P_3 as a Byzantine player. In that case P_1 and P_2 are given additional MAC keys to verify the shares coming from P_3.

The following result establishes the correctness of the protocol.

Theorem 3. *The protocol $\mathbf{\Pi^{CMP}_{Fair}}$ is U_w^{NF}-independent for $w \in \{1, 2\}$ and hence correct.*

Proof. We should recall that the deviations of P_1 and P_2 are similar. Thus for simplicity, here, we only consider the deviations of P_1.

In fail-stop setting, if P_1 aborts early and the round in which he aborts is less than j, according to Gordon et al.'s protocol, P_2 will output 0 and conclude that $i \leq j$. When $i > j$, it is the situation when P_2 is deceived by P_1. However, our protocol is designed in such a way that if P_1 has chosen abort in any round before r, P_2 will output $\perp$ and does not conclude anything. Thus, P_1 can not deceive P_2 by early abort. There is no incentive for P_1 to abort in a round $l > r$, as P_2 has already determined the output in some earlier iteration.

In case of Byzantine setting, P_1 can send arbitrary shares to both P_2 and P_3, so that P_2 will finally compute a wrong function. But since each share is signed by the dealer, no one can send an arbitrary share to the other. Another important deviation of P_1 in this setting is to swap the inputs. By swapping the inputs, P_1 can make P_2 compute a wrong function. As all the inputs came from the same dealer, there is no chance to catch this type of deviation by considering only the signature scheme. However, we consider signature with tagging. P_1 receives $a_1^1, a_2^1, \ldots, a_m^1$ and $(b_1^1, t_1^b), (b_2^1, t_2^b), \ldots, (b_m^1, t_m^b)$ and MAC key k_a. Similarly P_2 is given $(a_1^2, t_1^a), (a_2^2, t_2^a), \ldots, (a_m^2, t_m^a)$ and $b_1^2, b_2^2, \ldots, b_m^2$ and MAC key k_b. After receiving the share in the round l from P_1, if $Vrfy_{k_b}(l \parallel b_l^1, t_l^b) = 0$, then P_2 halts. Similar checking is done by P_3 as well. Thus, by input swapping no one can make the other believe in a wrong output.

Thus, assuming P_1 has $U_1^{NF} > U_1^{TT}$, the mechanism is designed in such a way that it becomes U_1^{NF} independent and hence correct. Proceeding in the same way for P_2, we can prove the U_2^{NF} independence. □

Now we are in a position to establish fairness of $\mathbf{\Pi^{CMP}_{Fair}}$.

Theorem 4. *Provided $\mathcal{R}_1$ and P_3 always plays the game according to the suggested strategy, the protocol $\mathbf{\Pi^{CMP}_{Fair}}$ achieves fairness.*

Proof. Without loss of generality, let us assume that P_3 is following the suggested strategy and the player P_1 is deviating. The analysis when P_2 deviates is similar.

In this case, the reason for deviation is to get the function alone. In fail-stop as well as in Byzantine setting P_1 can abort in round l.

P_1 may choose three types of abort in round l.

1) It may not send its share to only P_2.
2) It may not send its share to only P_3.
3) It may not send its share to both P_2 and P_3.

If P_1 does not send its share to P_2, then P_2 will not send its share to P_3. As a result the protocol will be terminated without producing any result either for P_1 or P_2. Similarly, if P_1 does not send its share to P_3, according to the protocol P_3 will output $\perp$ to both the players. In the third case, the protocol will be terminated from the beginning of the round l. Thus, there is no incentive for P_1 to abort early in the motivation to get the secret alone. □

Theorem 5. *Provided $\mathcal{R}_1$, $0 < \gamma < 1$, $\epsilon \leq \delta \leq U_w^{TN} - U_w^{TT}$, and $\gamma U_w^{TN} + (1-\gamma)U_w^{NN} < U_w^{TT}$ for all $w \in \{1, 2\}$, $\mathbf{\Pi^{CMP}_{Fair}}$ achieves strict Nash equilibrium.*

Proof. In Theorem 2, we proved that provided $\mathcal{R}_1$, $0 < \gamma < 1$, $\epsilon \leq \delta \leq U_w^{TN} - U_w^{TT}$ and $\gamma U_w^{TN} + (1-\gamma)U_w^{NN} < U_w^{TT}$ for all $w \in \{1, 2\}$, P_3 can not increase his utility value by deviating from the suggested strategy. We also proved that provided $\mathcal{R}_1$ and given that P_3 follows the protocol, neither P_1 nor P_2 can increase his utility value beyond U_w^{TT} (utility when each player follows the suggested strategy) by deviating from the suggested strategy (Theorem 4). In other words,

$$u_w(\sigma'_w, \overrightarrow{\sigma}_{-w}) < u_w(\overrightarrow{\sigma}),$$

which concludes the proof. □

3.3 Fairness analysis of $\Pi^{\mathbf{CMP}}$ when players have unequal domain size

As discussed in [10, Section 3.2], when the domain sizes of the players are unequal, the analysis in the non-rational setting does not change. It is easy to see from our analysis of Section 3.1 that even in the rational setting, we can carry out an analogous calculation to conclude that the protocol is U^{NF}-dependent and hence not fair.

4 Secure Two-Party Computation for Functions involving Embedded XOR with Rational Players

In this section, we first describe the embedded XOR problem proposed by Gordon et al. [9]. We, then, will show how fairness condition is affected in the presence of the rational players having the preferences $\mathcal{R}_1$ (refer to subsection 2.6). Let us denote two players by P_1 and P_2. Player P_1 is given an ordered list $\{x_1, x_2, x_3\}$ and P_2 is given an ordered list $\{y_1, y_2\}$. P_1 randomly chooses the input from the ordered list and sends to the dealer (the trusted party in the *hybrid model*). P_2 also randomly chooses the input from his list and delivers to the dealer. Dealer calculates the function. For convenience, we here recall the table for f given in [9].

	y_1	y_2
x_1	0	1
x_2	1	0
x_3	1	1

The function can be described as, for $w \in \{1, 2\}$

$$f(x_i, y_j) = \begin{cases} 1 & \text{if } i \neq j; \\ 0 & \text{if } i = j. \end{cases} \tag{5}$$

As described in [9], the protocol proceeds in a series of M iterations, where $M = \omega(\gamma^{-1} \log \lambda)$, λ being the security parameter. Let x and y denote the inputs from P_1 and P_2 respectively. The dealer chooses the revelation round r according to geometric distribution with parameter γ. The dealer then creates two sequences $\{a_l\}$ and $\{b_l\}$, $l = 1, 2, \ldots, M$, as follows.

$$\text{For } l \geq r, \qquad a_l = b_l = f(x, y).$$

$$\text{For } l < r, \quad a_l = f(x, \hat{y}), \quad b_l = f(\hat{x}, y),$$

where $\hat{x}$ (or $\hat{y}$) is a random value of x (or y) chosen by the dealer.

Next, the dealer splits the secret a_l into the shares a_l^1 and a_l^2, and the secret b_l into the shares b_l^1 and b_l^2, so that $a_l = a_l^1 \oplus a_l^2$ and $b_l = b_l^1 \oplus b_l^2$, and gives the shares $\{(a_l^1, b_l^1)\}$ to P_1 and the shares $\{(a_l^2, b_l^2)\}$ to P_2. In each round l, P_2 sends a_l^2 to P_1, who, in turn sends b_l^1 to P_2. P_1 and P_2 both learns the output value $f(x, y)$ in iteration r, unlike the millionaires' problem. The algorithm for the functionality share generation in fail-stop setting is revisited in Algorithm 5. Here we assume that the dealer who will distribute the shares is honest and can compute the function described in Equation (5). The protocol for computing f is described in Algorithm 6.

Inputs:
1 x from P_1 and y from P_2. If one of the received input is not in the correct domain, then both the parties are given $\perp$.
Computation:
The dealer does the following:
2 Chooses r according to a geometric distribution $\mathcal{G}(\gamma)$ with parameter γ. Here r is the revelation round, i.e., the round in which the value of f is either $(0,0)$ or $(1,1)$.
3 For $l < r$, sets $a_l = f(x, \hat{y})$ and $b_l = f(\hat{x}, y)$.
4 For $l \geq r$, sets $a_l = a_r$ and $b_l = b_r$.
5 For $l \in \{1, \ldots, M\}$, chooses a_l^1 randomly from $\{0,1\}^2$, and sets $a_l^2 = a_l^1 \oplus a_l$.
6 For $l \in \{1, \ldots, M\}$, chooses b_l^1 randomly from $\{0,1\}^2$, and sets $b_l^2 = b_l^1 \oplus b_l$.
Output:
7 The dealer prepares a list $list_w$ of shares for each party P_w, where $w \in \{1, 2\}$ such that
P_1 receives the values of $a_1^1, a_2^1, \ldots, a_M^1$ and $b_1^1, b_2^1, \ldots, b_M^1$.
P_2 receives the values of $a_1^2, a_2^2, \ldots, a_M^2$ and $b_1^2, b_2^2, \ldots, b_M^2$.

Algorithm 5: ShareGen2

Inputs:
1 P_1 obtains $a_1^1, a_2^1, \ldots, a_M^1$ and $b_1^1, b_2^1, \ldots, b_M^1$.
2 P_2 obtains $a_1^2, a_2^2, \ldots, a_M^2$ and $b_1^2, b_2^2, \ldots, b_M^2$.
Computation:
There are M number of iterations. In each iteration $l \in \{1, 2, \ldots, M\}$ do:
3 P_2 sends a_l^2 to P_1 and P_1 computes $a_l = a_l^1 \oplus a_l^2$.
4 P_1 sends b_l^1 to P_2 and P_2 computes $b_l = b_l^1 \oplus b_l^2$.
Output:
5 If P_2 aborts in round l, i.e., does not send its share at that round and $l \leq r$, P_1 outputs $a_{l-1} = f(x, \hat{y})$. If $l > r$, P_1 has already determined the output in some earlier iteration. Thus it outputs that value.
6 If P_1 aborts in round l, i.e., P_1 computes its output and does not send its share at that round and $l \leq r$, P_2 outputs $b_l = f(\hat{x}, y)$. If $l > r$, P_2 has already determined the output in some earlier iteration. Thus it outputs that value.

Algorithm 6: $\Pi^{\mathbf{CEP2}}$

The algorithms in the Byzantine setting are the same as those in the fail-stop setting except some additional steps. In

Byzantine setting, the shares are signed by the dealer. The signing message distribution procedure is same as Section 3.

4.1 Π^{CEP2} is not fair when players are rational

In this subsection, we analyze the fairness condition of the function in rational setting. We assume that the players, P_1 and P_2 have the preferences $\mathcal{R}_1$.

4.1.1 *Early abort by P_2*

Let us first assume that P_2 be corrupted by a probabilistic polynomial time adversary$\mathcal{A}$ and chooses to abort in the round $l \leq r$. Let U_2 be the utility of P_2 when he aborts. We have two cases depending on P_2's choice of y.

Case 1: $y = y_1$ Thus, $\Pr(b_{l-1} = 0|y = y_1) = \Pr(\hat{x} = x_1) = \frac{1}{3}$ and $\Pr(b_{l-1} = 1|y = y_1) = \Pr(\hat{x} \in \{x_2, x_3\}) = \frac{2}{3}$.

Under this case, three different subcases are possible depending on P_1's choice of x.

Subcase 1.(a): $x = x_1$. Now, $\Pr(a_{l-1} = 0|x = x_1) = \Pr(\hat{y} = y_1) = \frac{1}{2}$ and $\Pr(a_{l-1} = 1|x = x_1) = \Pr(\hat{y} = y_2) = \frac{1}{2}$. The following table enumerates the different possibilities for U_2 when $x = x_1$ and $y = y_1$.

(a_{l-1}, b_{l-1})	U_2	Probability
(0,0)	U_2^{TT}	$\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$
(0,1)	U_2^{NT}	$\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$
(1,0)	U_2^{TN}	$\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$
(1,1)	U_2^{NN}	$\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$

Thus, $E[U_2|(x_1, y_1)] = \Big[\frac{1}{6}(U_2^{TN} + U_2^{TT}) + \frac{1}{3}(U_2^{NT} + U_2^{NN})\Big]$.

Subcase 1.(b): $x = x_2$. Now, $\Pr(a_{l-1} = 0|x = x_2) = \Pr(\hat{y} = y_2) = \frac{1}{2}$ and $\Pr(a_{l-1} = 1|x = x_2) = \Pr(\hat{y} = y_1) = \frac{1}{2}$.

The following table enumerates the different possibilities for U_2 when $x = x_2$ and $y = y_1$.

(a_{l-1}, b_{l-1})	U_2	Probability
(0,0)	U_2^{NN}	$\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$
(0,1)	U_2^{TN}	$\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$
(1,0)	U_2^{NT}	$\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$
(1,1)	U_2^{TT}	$\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$

Thus, $E[U_2|(x_2, y_1)] = \Big[\frac{1}{6}(U_2^{NN} + U_2^{NT}) + \frac{1}{3}(U_2^{TN} + U_2^{TT})\Big]$.

Subcase 1.(c): $x = x_3$. In this case, P_1 knows the output with certainty. That means, $\Pr(a_{l-1} = 0|x = x_3) = 0$ and $\Pr(a_{l-1} = 1|x = x_3) = 1$.

The following table enumerates the different possibilities for U_2 when $x = x_3$ and $y = y_1$.

(a_{l-1}, b_{l-1})	U_2	Probability
(0,0)	U_2^{NN}	$0 \cdot \frac{1}{3} = 0$
(0,1)	U_2^{TN}	$0 \cdot \frac{2}{3} = 0$
(1,0)	U_2^{NT}	$1 \cdot \frac{1}{3} = \frac{1}{3}$
(1,1)	U_2^{TT}	$1 \cdot \frac{2}{3} = \frac{2}{3}$

Thus, $E[U_2|(x_3, y_1)] = \frac{1}{3}U_2^{NT} + \frac{2}{3}U_2^{TT}$.

Now, combining all three subcases, we get

$$\begin{aligned}
E[U_2|y_1] &= E[U_2|(x_1, y_1)] \cdot \Pr(x = x_1) \\
&\quad + E[U_2|(x_2, y_1)] \cdot \Pr(x = x_2) \\
&\quad + E[U_2|(x_3, y_1)] \cdot \Pr(x = x_3) \\
&= \Big[\frac{1}{6}(U_2^{TN} + U_2^{TT}) + \frac{1}{3}(U_2^{NT} + U_2^{NN})\Big] \cdot \frac{1}{3} \\
&\quad + \Big[\frac{1}{6}(U_2^{NN} + U_2^{NT}) + \frac{1}{3}(U_2^{TN} + U_2^{TT})\Big] \cdot \frac{1}{3} \\
&\quad + \Big[\frac{1}{3}U_2^{NT} + \frac{2}{3}U_2^{TT}\Big] \cdot \frac{1}{3} \\
&= \frac{1}{18}\Big[3U_2^{TN} + 7U_2^{TT} + 3U_2^{NN} + 5U_2^{NT}\Big].
\end{aligned}$$

If the above expression is greater than U_2^{TT}, P_2 aborts early, otherwise he plays the game.

Case 2: $y = y_2$ The analysis is similar and we obtain the same expression for $E[U_2|y_2]$. More specifically, we have the following observation.

Subcase 2.(a): $x = x_1$. The analysis is exactly identical to Subcase 1.(b).

Subcase 2.(b): $x = x_2$. The analysis is exactly identical to Subcase 1.(a).

Subcase 2.(c): $x = x_3$. The analysis is exactly identical to Subcase 1.(c).

4.1.2 *Early abort by P_1*

Now, we consider the aborting of P_1. We assume that there is a probabilistic polynomial time adversary $\mathcal{A}$ who corrupts P_1 and makes P_1 to choose abort in round l. Let U_1 be the utility of P_1 when he aborts. We have three cases depending on P_1's choice of x.

Case 1: $x = x_1$ We have $\Pr(a_l = 0|x = x_1) = \Pr(\hat{y} = y_1) = \frac{1}{2}$ and $\Pr(a_l = 1|x = x_1) = \Pr(\hat{y} = y_2) = \frac{1}{2}$, for $l < r$. Note that for $l = r$, P_1 will abort after receiving the exact value of y. Hence, in case of $y = y_1$,

$$\Pr(a_r = 0|(x_1, y_1)) = 1, \qquad \Pr(a_r = 1|(x_1, y_1)) = 0;$$

in case of $y = y_2$,

$$\Pr(a_r = 0|(x_1, y_2)) = 0, \qquad \Pr(a_r = 1|(x_1, y_2)) = 1.$$

Subcase 1.(a): $y = y_1$. Now, we have $\Pr(b_l = 0|y = y_1) = \Pr(\hat{x} = x_1) = \frac{1}{3}$ and $\Pr(b_l = 1|y = y_1) = \Pr(\hat{x} \in \{x_2, x_3\}) = \frac{2}{3}$.

The following table enumerates the different possibilities for U_1 when $x = x_1$ and $y = y_1$.

(a_l, b_l)	U_1	Probability	
		$l < r$	$l = r$
(0,0)	U_1^{TT}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{1}{3} = (1-\gamma) \cdot \frac{1}{6}$	$\gamma \cdot 1 \cdot \frac{1}{3} = \gamma \cdot \frac{1}{3}$
(0,1)	U_1^{TN}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{2}{3} = (1-\gamma) \cdot \frac{1}{3}$	$\gamma \cdot 1 \cdot \frac{2}{3} = \gamma \cdot \frac{2}{3}$
(1,0)	U_1^{NT}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{1}{3} = (1-\gamma) \cdot \frac{1}{6}$	$\gamma \cdot 0 \cdot \frac{1}{3} = 0$
(1,1)	U_1^{NN}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{2}{3} = (1-\gamma) \cdot \frac{1}{3}$	$\gamma \cdot 0 \cdot \frac{2}{3} = 0$

Thus,

$$\begin{aligned}
& E[U_1|(x_1, y_1)] \\
= \ & (1-\gamma)\Big[\frac{1}{3}U_1^{TN} + \frac{1}{6}U_1^{TT} + \frac{1}{3}U_1^{NN} + \frac{1}{6}U_1^{NT}\Big] \\
& + \gamma\Big[\frac{2}{3}U_1^{TN} + \frac{1}{3}U_1^{TT}\Big] \\
= \ & \frac{(1+\gamma)}{6}\Big(2U_1^{TN} + U_1^{TT}\Big) + \frac{(1-\gamma)}{6}\Big(2U_1^{NN} + U_1^{NT}\Big).
\end{aligned}$$

Subcase 1.(b): $y = y_2$. Now, we have $\Pr(b_l = 0|y = y_2) = \Pr(\hat{x} = x_2) = \frac{1}{3}$ and $\Pr(b_l = 1|y = y_2) = \Pr(\hat{x} \in \{x_1, x_3\}) = \frac{2}{3}$.

The following table enumerates the different possibilities for U_1 when $x = x_1$ and $y = y_2$.

(a_l, b_l)	U_1	Probability	
		$l < r$	$l = r$
(0,0)	U_1^{NN}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{1}{3} = (1-\gamma) \cdot \frac{1}{6}$	$\gamma \cdot 0 \cdot \frac{1}{3} = 0$
(0,1)	U_1^{NT}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{2}{3} = (1-\gamma) \cdot \frac{1}{3}$	$\gamma \cdot 0 \cdot \frac{2}{3} = 0$
(1,0)	U_1^{TN}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{1}{3} = (1-\gamma) \cdot \frac{1}{6}$	$\gamma \cdot 1 \cdot \frac{1}{3} = \frac{1}{3}$
(1,1)	U_1^{TT}	$(1-\gamma) \cdot \frac{1}{2} \cdot \frac{2}{3} = (1-\gamma) \cdot \frac{1}{3}$	$\gamma \cdot 1 \cdot \frac{2}{3} = \frac{2}{3}$

$$\begin{aligned} E[U_1|(x_1, y_2)] &= (1-\gamma)\Big(\frac{1}{6}U_1^{TN} + \frac{1}{3}U_1^{TT} + \frac{1}{6}U_1^{NN} \\ &\quad + \frac{1}{3}U_1^{NT}\Big) + \gamma\Big(\frac{1}{3}U_1^{TN} + \frac{2}{3}U_1^{TT}\Big) \\ &= \frac{(1+\gamma)}{6}\Big(U_1^{TN} + 2U_1^{TT}\Big) \\ &\quad + \frac{(1-\gamma)}{6}\Big(U_1^{NN} + 2U_1^{NT}\Big). \end{aligned}$$

Now, combining all two subcases, we get

$$\begin{aligned} & E[U_1|x_1] \\ &= E[U_1|(x_1, y_1)] \cdot \Pr(y = y_1) + E[U_1|(x_1, y_2)] \cdot \Pr(y = y_2) \\ &= \Big[\tfrac{(1+\gamma)}{6}\Big(2U_1^{TN} + U_1^{TT}\Big) + \tfrac{(1-\gamma)}{6}\Big(2U_1^{NN} + U_1^{NT}\Big)\Big] \cdot \tfrac{1}{2} \\ &\quad + \Big[\tfrac{(1+\gamma)}{6}\Big(U_1^{TN} + 2U_1^{TT}\Big) + \tfrac{(1-\gamma)}{6}\Big(U_1^{NN} + 2U_1^{NT}\Big)\Big] \cdot \tfrac{1}{2} \\ &= \tfrac{1+\gamma}{4}\Big(U_1^{TN} + U_1^{TT}\Big) + \tfrac{1-\gamma}{4}\Big(U_1^{NN} + U_1^{NT}\Big). \end{aligned}$$

If the above expression is greater than U_1^{TT}, P_1 chooses abort.

Case 2: $x = x_2$ The analysis is similar and we obtain the same expression for $E[U_1|x_2]$. More specifically, we have the following observation.

Subcase 2.(a): $y = y_1$. The analysis is exactly identical to Subcase 1.(b).

Subcase 2.(b): $y = y_2$. The analysis is exactly identical to Subcase 1.(a).

Case 3: $x = x_3$ In this case, P_1 has no incentive to play as he knows in certainty that the output should be 1. For any $l \leq r$, P_1 always has expected utility $\Big[\frac{2}{3}U_1^{TT} + \frac{1}{3}U_1^{TN}\Big]$, which is always greater than U_1^{TT}. Thus, if P_1 chooses x_3, he always aborts early and fairness can not be achieved.

4.1.3 Summary of the analysis

From the above analysis, it is clear that aborting of P_2 does not affect fairness. If P_2 aborts and $l \leq r$, then no one obtains the output. However, if $l > r$, then both obtain the output. Contrary to this, aborting of P_1 affects fairness, as he computes the output first from the input received from P_2. When $x = x_3$, P_1 should have no incentive to continue the game as he knows the output with certainty. Thus, we observe that Gordon et al.'s protocol $\mathbf{\Pi^{CEP2}}$ of [9], [10], that was designed for non-rational setting, is no longer fair in the rational setting.

Theorem 6. *The protocol $\mathbf{\Pi^{CEP2}}$ can not achieve fairness with rational players.*

4.2 How to make $\mathbf{\Pi^{CEP2}}$ fair when players are rational

In this subsection we suggest a variant of Gordon et al.'s protocol with fairness in the presence of a rational adversary. Here, we only modify the step 6 of Algorithm 6: $\mathbf{\Pi^{CEP2}}$, and call the resulting protocol $\mathbf{\Pi^{CEP2}_{Fair}}$. When P_1 aborts in any round l, instead of obtaining $f(\hat{x}, y)$, P_2 outputs 1. Every other steps are remain same. We now prove the fairness of the protocol.

4.2.1 Early abort by P_2

The analysis in this case is exactly identical to Section 4.1.1. Thus, for fairness, we need to ensure that

$$\frac{1}{18}\Big[3U_2^{TN} + 7U_2^{TT} + 3U_2^{NN} + 5U_2^{NT}\Big] \leq U_2^{TT},$$

i.e.,

$$U_2^{TT} \geq \frac{1}{11}\Big[3U_2^{TN} + 3U_2^{NN} + 5U_2^{NT}\Big]. \tag{6}$$

4.2.2 Early abort by P_1

Now, we discuss each case one by one.

Case 1: $x = x_1$ We have $\Pr(a_l = 0|x = x_1) = \Pr(\hat{y} = y_1) = \frac{1}{2}$ and $\Pr(a_l = 1|x = x_1) = \Pr(\hat{y} = y_2) = \frac{1}{2}$, for $l < r$. Note that for $l = r$, P_1 will abort after receiving the exact value of y. Hence, in case of $y = y_1$,

$$\Pr(a_r = 0|(x_1, y_1)) = 1, \qquad \Pr(a_r = 1|(x_1, y_1)) = 0;$$

in case of $y = y_2$,

$$\Pr(a_r = 0|(x_1, y_2)) = 0, \qquad \Pr(a_r = 1|(x_1, y_2)) = 1.$$

Subcase 1.(a): $y = y_1$. Now, we have $\Pr(b_l = 0|y = y_1) = 0$ and $\Pr(b_l = 1|y = y_1) = 1$.

The following table enumerates the different possibilities for U_1 when $x = x_1$ and $y = y_1$.

(a_l, b_l)	U_1	Probability	
		$l < r$	$l = r$
(0,0)	U_1^{TT}	$(1-\gamma) \cdot \frac{1}{2} \cdot 0 = 0$	$\gamma \cdot 1 \cdot 0 = 0$
(0,1)	U_1^{TN}	$(1-\gamma) \cdot \frac{1}{2} \cdot 1 = (1-\gamma) \cdot \frac{1}{2}$	$\gamma \cdot 1 \cdot 1 = \gamma \cdot 1$
(1,0)	U_1^{NT}	$(1-\gamma) \cdot \frac{1}{2} \cdot 0 = (1-\gamma) \cdot 0$	$\gamma \cdot 0 \cdot 0 = 0$
(1,1)	U_1^{NN}	$(1-\gamma) \cdot \frac{1}{2} \cdot 1 = (1-\gamma) \cdot \frac{1}{2}$	$\gamma \cdot 0 \cdot 1 = 0$

Thus,

$$\begin{aligned} E[U_1|(x_1, y_1)] &= (1-\gamma)\Big[\frac{1}{2}U_1^{TN} + \frac{1}{2}U_1^{NN}\Big] + \gamma\Big[U_1^{TN}\Big] \\ &= \frac{(1+\gamma)}{2}\Big(U_1^{TN}\Big) + \frac{(1-\gamma)}{2}\Big(U_1^{NN}\Big). \end{aligned}$$

Subcase 1.(b): $y = y_2$. Now, we have $\Pr(b_l = 0|y = y_2) = 0$ and $\Pr(b_l = 1|y = y_2) = 1$.

The following table enumerates the different possibilities for U_1 when $x = x_1$ and $y = y_2$.

(a_l, b_l)	U_1	Probability	
		$l < r$	$l = r$
(0,0)	U_1^{NN}	$(1-\gamma) \cdot \frac{1}{2} \cdot 0 = (1-\gamma) \cdot 0$	$\gamma \cdot 0 \cdot 0 = 0$
(0,1)	U_1^{NT}	$(1-\gamma) \cdot \frac{1}{2} \cdot 1 = (1-\gamma) \cdot \frac{1}{2}$	$\gamma \cdot 0 \cdot 1 = 0$
(1,0)	U_1^{TN}	$(1-\gamma) \cdot \frac{1}{2} \cdot 0 = (1-\gamma) \cdot 0$	$\gamma \cdot 1 \cdot 0 = 0$
(1,1)	U_1^{TT}	$(1-\gamma) \cdot \frac{1}{2} \cdot 1 = (1-\gamma) \cdot \frac{1}{2}$	$\gamma \cdot 1 \cdot 1 = \gamma$

$$\begin{aligned} E[U_1|(x_1, y_2)] &= (1-\gamma)\Big(\frac{1}{2}U_1^{TT} + \frac{1}{2}U_1^{NT}\Big) + \gamma\Big(U_1^{TT}\Big) \\ &= \frac{(1+\gamma)}{2}\Big(U_1^{TT}\Big) + \frac{(1-\gamma)}{2}\Big(U_1^{NT}\Big). \end{aligned}$$

Now, combining all two subcases, we get

$$\begin{aligned}E[U_1|x_1] &= E[U_1|(x_1,y_1)]\cdot\Pr(y=y_1)\\&\quad+E[U_1|(x_1,y_2)]\cdot\Pr(y=y_2)\\&= \Big[\frac{(1+\gamma)}{2}\Big(U_1^{TN}\Big)+\frac{(1-\gamma)}{2}\Big(U_1^{NN}\Big)\Big]\cdot\frac{1}{2}\\&\quad+\Big[\frac{(1+\gamma)}{2}\Big(U_1^{TT}\Big)+\frac{(1-\gamma)}{2}\Big(U_1^{NT}\Big)\Big]\cdot\frac{1}{2}\\&= \frac{(1+\gamma)}{4}\Big(U_1^{TN}+U_1^{TT}\Big)\\&\quad+\frac{(1-\gamma)}{4}\Big(U_1^{NN}+U_1^{NT}\Big).\end{aligned}$$

If the above expression is greater than U_1^{TT}, P_1 chooses abort. Thus, for fairness, we need to ensure that $U_1^{TT} \geq \frac{(1+\gamma)}{4}\Big(U_1^{TN}+U_1^{TT}\Big)+\frac{(1-\gamma)}{4}\Big(U_1^{NN}+U_1^{NT}\Big)$, i.e.,

$$\gamma \leq \frac{3U_1^{TT}-U_1^{TN}-U_1^{NN}-U_1^{NT}}{U_1^{TN}+U_1^{TT}-U_1^{NN}-U_1^{NT}}. \quad (7)$$

Case 2: $x = x_2$ The analysis is similar and we obtain the same expression for $E[U_1|x_2]$. More specifically, we have the following observation.

Subcase 2.(a): $y = y_1$. The analysis is exactly identical to Subcase 1.(b).

Subcase 2.(b): $y = y_2$. The analysis is exactly identical to Subcase 1.(a).

Case 3: $x = x_3$ When $x = x_3$, P_1 will abort as he knows the output with certainty. In this case, he needs no help from P_2 to compute the function. However, when P_1 chooses to abort, P_2 outputs 1. Thus, for $x = x_3$, both get the correct output of the function. The utility for both the player is U_w^{TT}, $w \in \{1,2\}$. Hence, the fairness condition in rational setting is always maintained.

4.2.3 Fairness condition

From the above analysis, we can state the following result.

Theorem 7. *Provided $\mathcal{R}_1$, $U_2^{TT} \geq \frac{1}{11}\Big[3U_2^{TN}+3U_2^{NN}+5U_2^{NT}\Big]$, and*

$$0 < \gamma \leq \frac{3U_1^{TT}-U_1^{TN}-U_1^{NN}-U_1^{NT}}{U_1^{TN}+U_1^{TT}-U_1^{NN}-U_1^{NT}},$$

the protocol $\mathbf{\Pi}^{\mathbf{CEP2}}_{\mathbf{Fair}}$ achieves fairness.

Proof. The proof follows from Equations (6) and (7). From the condition $\gamma \leq \frac{3U_1^{TT}-U_1^{TN}-U_1^{NN}-U_1^{NT}}{U_1^{TN}+U_1^{TT}-U_1^{NN}-U_1^{NT}}$, it is easy to see that the natural restriction $\gamma \leq 1$ always holds. □

Note that $\gamma > 0$ naturally implies that the numerator $3U_1^{TT}-U_1^{TN}-U_1^{NN}-U_1^{NT}$ is also > 0, i.e.,

$$(U_1^{TT}-U_1^{NN})+(U_1^{TT}-U_1^{NT}) > (U_1^{TN}-U_1^{TT}). \quad (8)$$

In Equation (8), all the three terms within the parentheses are non-negative according to $\mathcal{R}_1$.

Again, the condition $U_2^{TT} \geq \frac{1}{11}\Big[3U_2^{TN}+3U_2^{NN}+5U_2^{NT}\Big]$ can be re-written as

$$3(U_2^{TT}-U_2^{NN})+5(U_2^{TT}-U_2^{NT}) \geq 3(U_2^{TN}-U_2^{TT}). \quad (9)$$

If the utilities are symmetric, i.e., if $U_1^{xy} = U_2^{xy}$, then Equation (8) implies Equation (9), and hence we need one less condition. The following corollary is immediate.

Corollary 1. *Provided $\mathcal{R}_1$, and*

$$0 < \gamma \leq \frac{3U^{TT}-U^{TN}-U^{NN}-U^{NT}}{U^{TN}+U^{TT}-U^{NN}-U^{NT}},$$

the protocol $\mathbf{\Pi}^{\mathbf{CEP2}}_{\mathbf{Fair}}$ with symmetric utilities achieves fairness.

4.3 Fairness analysis of $\mathbf{\Pi}^{\mathbf{CEP2}}$ when players have equal domain sizes

In rational setting, the analysis of the original $\mathbf{\Pi}^{\mathbf{CEP2}}$ protocol [9], [10], is exactly the same as in Section 4.1 except that the cases corresponding to x_3 would not be there. In this situation, the protocol need not be modified. In order to maintain fairness, we keep the original steps of [9], [10], as in Algorithm 6, and Theorem 7 guarantees fairness. Note that the fairness condition is the same for unequal as well as equal domain sizes.

Theorem 8. *Provided $\mathcal{R}_1$, $U_2^{TT} \geq \frac{1}{11}\Big[3U_2^{TN}+3U_2^{NN}+5U_2^{NT}\Big]$, and*

$$0 < \gamma \leq \frac{3U_1^{TT}-U_1^{TN}-U_1^{NN}-U_1^{NT}}{U_1^{TN}+U_1^{TT}-U_1^{NN}-U_1^{NT}},$$

the protocol $\mathbf{\Pi}^{\mathbf{CEP2}}_{\mathbf{Fair}}$ achieves strict Nash equilibrium.

Proof. Provided $\mathcal{R}_1$ and $U_2^{TT} \geq \frac{1}{11}\Big[3U_2^{TN}+3U_2^{NN}+5U_2^{NT}\Big]$, P_2 can not increase his utility beyond U_2^{TT} by choosing deviating strategy (here early abort, Section 4.2.1). Similarly, provided $\mathcal{R}_1$, and

$$0 < \gamma \leq \frac{3U_1^{TT}-U_1^{TN}-U_1^{NN}-U_1^{NT}}{U_1^{TN}+U_1^{TT}-U_1^{NN}-U_1^{NT}},$$

P_1 can not increase his utility beyond U_1^{TT} by choosing deviating strategy (here, early abort, Section 4.2.2). In other words, for every player P_w

$$u_w(\sigma'_w, \overrightarrow{\sigma}_{-w}) < u_w(\overrightarrow{\sigma}).$$

This concludes the proof. □

5 Conclusion

In this paper, we revisit the 'greater than' function proposed by Gordon et al. [9], [10] as well as the problem of [10] that is an instance of the embedded XOR class. We show that in rational domain, none of the above two remains fair and then we propose variants that achieve fairness when the players are rational.

In 2011 Asharov et al. [2] showed the impossibility of the secure computation in two party setting in the rational domain. However, in 2012 Groce and Katz [12] proved that the results of Asharov et al. is correct for a specific function, specific input distribution and specific set of utilities. They [12] proposed a scheme for secure two party computation in rational setting for arbitrary function in incentive compatible settings. Their proposed protocol is an incomplete information game and thus achieves Bayesian strict Nash equilibrium. Also, the dealer is not completely offline in their protocol. Contrary to this, our proposed protocols for 'greater than' as well as the embedded XOR functions are complete information games and achieve strict Nash equilibrium without the necessity of an online dealer.

References

[1] G. Asharov, Y. Lindell. Utility Dependence in Correct and Fair Rational Secret Sharing. *Journal of Cryptology.* **24**, 157–202, (2010).

[2] G. Asharov, R. Canetti, C. Hazay. Towards a Game Theoretic View of Secure Computation. EUROCRYPT 2011, LNCS **6632**, 426–445, (2011).

[3] G. Asharov, Y. Lindell, T. Rabin. A Full Characterization of Functions that Imply Fair Coin Tossing and Ramifications to Fairness. *Proceedings of the 10th theory of cryptography conference on Theory of Cryptography* TCC'13, 243–262, ACM Press, (2013).

[4] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. *Proc. ACM STOC 1988*, (STOC), 1–10, (1988).

[5] R. Canetti, A. Rosen Lecture 8, Cryptography and Game Theory, December 9, 2009, www.cs.tau.ac.il/ canetti/f09-materials/f09-scribe8.pdf

[6] D. Chaum, C. Crépeau, I. Damgard. Multi-party unconditionally secure protocols. *Proc. ACM STOC 1988*, (STOC), 11–19, (1988).

[7] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract) *Proceedings of the 18th Annual ACM symposium on Theory of Computing* (STOC), 364–369, ACM Press, (1986).

[8] G. Fuchsbauer, J. Katz, D. Naccache. Efficient Rational Secret Sharing in Standard Communication Networks. *Proceedings of the 7th Conference on Theory of Cryptography*, 419–436, Springer-Verlag, (2010).

[9] S. D. Gordon, C. Hazay, J. Katz, Y. Lindell. Complete Fairness in Secure Two-Party Computation. *Proceedings of 40th Annual ACM symposium on Theory of Computing* (STOC), 413–422, ACM Press, (2008)

[10] S. D. Gordon, C. Hazay, J. Katz, Y. Lindell. Complete Fairness in Secure Two-Party Computation. *Journal of the ACM (JACM)* **58**, Issue 6, December 2011.

[11] S. D. Gordon, J. Katz. Rational Secret Sharing, Revisited. *Security and Cryptography for Networks, Springer*, 229–241, (2006).

[12] A. Groce, J. Katz. Fair computation with rational players. EUROCRYPT 2012, LNCS **7237**, 81–98, Springer (2012).

[13] J. Halpern, V. Teague. Rational secret sharing and multiparty computation: extended abstract. *Proceedings of the 36th Annual ACM symposium on Theory of computing*, 623–632 (2004).

[14] G. Kol, M. Naor. Games for exchanging information. *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 423–432, (2008).

[15] Y. Lindell. Composition of Secure Multi-Party Protocols, A Comprehensive study. Springer-Verlag, Berlin, (2003).

[16] A. Shamir. How to share a secret. *Communications of the ACM* **22** 612–613, (1979).

[17] A. Lysyanskaya, A. Segal. Rational Secret Sharing with Side Information in Point-to-Point Networks via Time Delayed Encryption. IACR Cryptology ePrint Archive : Report 2010/540, (2010).

[18] A. Lysyanskaya, N. Triandopoulos. Rationality and Adversarial Behavior in Multi-party Computation. *Advances in Cryptology - CRYPTO'06*, 180–197, (2006).

[19] M. Nojoumian, D. R. Stinson. Socio-Rational Secret Sharing as a New Direction in Rational Cryptography. 3rd Conference on Decision and Game Theory for Security 2012 (GameSec'12), LNCS, **7638**, 18–37, (2012).

[20] John von Neumann, Oskar Morgenstern. Theory of Games and Economic Behavior. Princeton University Press, 1944.

[21] S. J. Ong, D. V. Parkes, A. Rosen, S. Vadhan. Fairness with an Honest Minority and a Rational Majority. *Proceedings of the 6th Conference on Theory of Cryptography*, 36–53, Springer-Verlag, (2009).

[22] A. C. Yao. Protocols for secure computations. *23rd Annual Symposium on Foundations of Computer Science* (FOCS), 160–164, (1982).