Indian Institute of Management Calcutta

Working Paper Series

WPS No. 790
November 2016

On the Fairness of Groce–Katz's Protocol for Rational Players

Arpita Maitra
Research Assistant
Indian Institute of Management Calcutta, D. H. Road, Joka, P.O. Kolkata 700 104
Email: arpita76b@gmail.com

Asim K. Pal
Professor
Indian Institute of Management Calcutta
D. H. Road, Joka, P.O. Kolkata 700 104
http://facultylive.iimcal.ac.in/workingpapers

On the Fairness of Groce–Katz's Protocol for Rational Players*

Arpita Maitra and Asim K. Pal
Indian Institute of Management Calcutta
Email: arpita76b@gmail.com, asim@iimcal.ac.in

Abstract

In Eurocrypt 2012, Groce and Katz provided a mathematical description about 'incentive compatible' setting in the context of fair two party computation with rational players. They showed, how by modifying the utility values, 'incentive incompatible' setting can be converted into 'incentive compatible' setting for an XOR function. In this paper, we try to understand, whether by modification of the utility values, 'incentive incompatible' setting could always be converted into 'incentive compatible' setting for any function. In this direction, we observe two distinct classes of functions which show 'incentive incompatibility' for any value of utilities assuming certain guessing strategies and input distribution. One class includes all functions without an embedded XOR and other class has a specific function containing an embedded XOR. Such functions had been used to show the first fair two party secure computation with non-rational players (Gordon et al., STOC 2008). Our observations help to understand the structure of such 'incentive incompatible' functions.

Keywords: Secure two party computation, Embedded XOR, Rational players, Incentive compatible

1 Introduction

In the last decade, a significant effort has been made to relate game theory and cryptography. Cryptography deals with the worst case scenario and tries to analyse the robustness of a system against the malicious behaviour of an adversary. On the other hand, in game theory, a mechanism is designed considering rational behaviour of the players. In rational domain there is no concept of trust. Rational players can never be classified as 'good' nor 'bad'. They participate in the game with a motivation to maximize their utility. In cryptography,

*The current work is supported by the project "Information Security and Quantum Cryptography: Study of Secure Multi-Party Computation in Quantum Domain and Applications" at IIM Calcutta.

one may consider this as a special type of attack vector. However, this does not impose any special restriction on an adversary, rather adds more flexibility.

In 2004 Halpern and Teague [11] proposed rational adversarial model in Shamir's secret sharing scheme [17]. This work is one of the initial efforts to connect cryptography and game theory. They proved that in the presence of rational adversary Shamir's secret sharing scheme achieves Nash equilibrium. This corresponds to the case when no player has any incentive to broadcast the share to others. Since then, a large number of literature [1, 2, 6, 7, 12, 14, 15] had been published in this domain.

Secure Multiparty Computation is one of the fundamental primitives of cryptography. In SMC, n number of parties wish to compute a function $f(x_1, x_2, \cdots, x_n)$ of their inputs $x_1, x_2, \cdots, x_n$ keeping the inputs secret from each other. SMC has wide application in online auction, negotiation, electronic voting etc. Since Yao's millionaire's problem [19], which is considered as the first attempt in the domain of SMC, a huge research interest has been drawn towards this area. Recently, the concept of rational adversarial model has been introduced [3, 10]. In [3], Asharov et al. showed the impossibility of computing a special class of functions with complete fairness if the players have a specific set of utilities. Further, in [10], Groce et al. claimed that the negative result had been achieved because Asharov et al. assumed incentive incompatible setting. Thus, no player has any incentive to compute the function even in the ideal world where everything is believed to be secure. However, they claimed that if the setting is incentive compatible, then any function can be computed with complete fairness in the presence of rational players. They showed that properly modifying the utility values the functions considered in [3] can be mapped into incentive compatible setting and thus the fairness can be guaranteed. However, it has not been studied that, whether modifying the utility values 'incentive incompatible' setting could be converted to 'incentive compatible' setting for any arbitrary functions. There could indeed be some functions which are 'incentive incompatible' by nature, i.e., modifying the utility values one can never convert the setting to an 'incentive compatible' one.

In this paper we identify two classes of functions for which some instances remain incentive incompatible for any value of utilities.

- One class consists of functions having no embedded XOR within them.
- Another class deals with a specific function having an embedded XOR.

Such functions are of importance in secure multiparty computation for the following reason. Since the negative result of Cleve [5], it had been conjectured that no function can be computed securely in two party setting with complete fairness. However, Gordon et al. [8, 9] presented the above mentioned functions and disproved the conjecture. It has been shown that for such functions secure two party computation is possible with complete fairness. The same classes of functions are studied in this work and we observe that such functions are incentive incompatible in nature.

2 Preliminaries

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

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 represents the output of the first party, say P_1. Similarly, f_λ^2 represents the 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 [9].

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 an input $x \in X_\lambda$ and party P_2 is holding 1^λ and an input $y \in Y_\lambda$, then the joint distribution of the outputs of the parties is statistically close to $(f_\lambda^1(x, y), f_\lambda^2(x, y))$ [9].

2.3 Ideal vs. Real world model

In *ideal world model* we assume that there is an incorruptible Trusted Third Party (TTP) who computes the function in behalf of P_1 and P_2. The parties 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. Instead, a protocol is executed to compute the functionality.

There exists another model, called the *hybrid world model*, which is often used as a powerful tool to prove the security of a protocol. In hybrid model there is an 'ideal functionality' ShareGen. One can think of this ideal functionality as a trusted third party (or dealer) who computes the function like *ideal world* and distributes the shares of the function's output like a dealer in the secret sharing scheme [17]. It is now proven that this 'ideal functionality ShareGen' can be replaced by sequential execution of a real-world protocol ρ that computes some functionality $\mathcal{G}$ under certain computational hardness assumption. Consider that ρ is a real world protocol which computes a functionality $\mathcal{G}$ securely and π is a protocol that securely computes a functionality $\mathcal{F}$ in $\mathcal{G}$-hybrid model. That is, $\mathcal{G}$ is computed according to the ideal model with abort. Then it has been shown in [4] that there exists a composed

two party protocol π^{ρ} which can securely compute $\mathcal{F}$. In [10], a protocol in rational setting is proposed assuming this hybrid model.

2.4 Rational secret sharing

In this subsection, we briefly describe what is meant by classical rational secret sharing. It proceeds in two phases:

1. share generation and distribution, and
2. secret reconstruction.

The dealer generates the shares from the secret and distributes among the players in the first phase. The dealer is assumed to be honest and can be online or offline. An online dealer remains available throughout the secret reconstruction protocol, whereas an offline dealer becomes unavailable after distributing the shares of the secret. Note that an online dealer is not practical as it repeatedly interacts with the players. In 2008, Kol and Naor [12] discussed rational secret sharing in the non-simultaneous channel model and in the presence of an offline dealer, under certain information theoretic setting. Almost all the subsequent works [2, 6, 15, 16] on rational secret sharing assumed the dealer to be offline.

Share generation and distribution: Consider that the dealer is online. At the beginning of each round, the shares are distributed to each player P_w. This might be the actual share with the probability γ or a fake one with probability $(1-\gamma)$. The value of γ is kept secret from the parties and is dependent on the utility values of the parties [11, 7]. Now consider the scenario with an offline dealer. In this case, the dealer distributes a list of shares to each party P_w. Among those one is of the actual secret s and the remaining are from fake secrets [12, 6, 15]. The position r of this actual share in the lists is not revealed to the players and is chosen according to a geometric distribution $G(\gamma)$, where the parameter γ in turn depends on the utility values of players. The dealer generates shares using Shamir's secret sharing scheme [17].

Secret Reconstruction: In the l-th round of communication, each player P_w (either simultaneously or non-simultaneously) broadcasts (or sends individually to each other player through synchronous, point-to-point channels) the share s_{wl} corresponding to that round. The shares are signed by the dealer. Hence, no player can provide false shares undetected and the only possible actions of a player are as follows.

1. send the message
2. or, remain silent.

The round, in which the shares of the actual secret are revealed and hence the secret is reconstructed, is called the revelation or definitive round. When the dealer is offline, players are made aware that they have crossed the revelation round by the reconstruction of an indicator (a bit in [12], a signal in [6]).

2.5 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 [10] 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.

2.6 Computation of a functionality with rational players

We define a *functionality computation with rational players* 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 players. We use the notations $\overrightarrow{\sigma}_{-w}$ and $(\sigma'_w, \overrightarrow{\sigma}_{-w})$ for $(\sigma_1, \ldots, \sigma_{w-1}, \sigma_{w+1}, \ldots, \sigma_n)$ and $(\sigma_1, \ldots, \sigma_{w-1}, \sigma'_w, \sigma_{w+1}, \ldots, \sigma_n)$ respectively. 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, i.e.,
   $\overrightarrow{o}(\Gamma, \overrightarrow{\sigma}) = (o_w = f, o_{-w} = \rightharpoonup)$
2. everybody correctly computes f, i.e., $\overrightarrow{o}(\Gamma, \overrightarrow{\sigma}) = (o_w = f, o_{-w} = f)$
3. nobody correctly computes f i.e. $\overrightarrow{o}(\Gamma, \overrightarrow{\sigma}) = (o_w = \rightharpoonup, o_{-w} = \rightharpoonup)$
4. others computes f correctly, while P_w does not, i.e.,
   $\overrightarrow{o}(\Gamma, \overrightarrow{\sigma}) = (o_w = \rightharpoonup, o_{-w} = f)$

The output, when a wrong value is computed, is denoted by $\rightharpoonup$ and o_{-w} indicates the output of a party other than P_w.

2.7 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 1. The notations used in [10] for utility functions are mentioned in the first bracket.

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

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

Table 1: Outcomes and Utilities for a function reconstruction mechanism with rational players in two party setting

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 $U_2(o_1, o_2)$
$o_1 = f$	$o_2 = f$	$U_1^{TT}(a_0)$	$U_2^{TT}(a_1)$
$o_1 = \neg$	$o_2 = \neg$	$U_1^{NN}(d_0)$	$U_2^{NN}(d_1)$
$o_1 = f$	$o_2 = \neg$	$U_1^{TN}(b_0)$	$U_2^{NT}(c_1)$
$o_1 = \neg$	$o_2 = f$	$U_1^{NT}(c_0)$	$U_2^{TN}(b_1)$

2.8 Fairness

In non-rational setting, the security of a protocol is analyzed by comparing what an adversary can achieve in a *real* world scenario to what it can do in an *ideal* one (the ideal one is secure by definition [8, 9, 13]). This is formalized by considering an ideal computation involving an incorruptible trusted party to whom all 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 ideal world computation.

A rational player, being selfish, desires an unfair outcome, i.e., it tries to compute the function on its own without any collaboration. Therefore, the basic aim of rational computation is to achieve fairness. According to Von Neumann and Morgenstern *expected utility theorem* [18], 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 [10] implicitly uses this expected utility theorem to derive its results. We also use the same approach and accordingly redefine fairness as follows.

Definition 1 *(Fairness) A function computation mechanism with rational players* $(\Gamma, \overrightarrow{\sigma})$ *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)*. Here* n' *is the number of possible outcomes.*

2.9 Correctness of the Output

Secure two party computation in *hybrid model* with rational players is the generalization of [7, 11, 12]. In [7, 11, 12], 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., neither 'good' nor 'bad'. They have no motivation to send incorrect inputs so that the functionality generating protocol outputs a wrong value. Rather they try to maximize their utility [10].

3 Revisiting the Protocol by Groce and Katz

In this section we revisit the protocol of [10]. The protocol has been designed towards computing an arbitrary function with complete fairness with the presence of rational players. According to their model, given a deterministic function $f : X \times Y \to \{0,1\}^* \times \{0,1\}^*$, two players P_1 and P_2 want to compute the function to their respective ends keeping the inputs ($x \in X$ for player P_1 and $y \in Y$ for player P_2) secret from each other. Here, it has been assumed that $|X| = |Y| = \lambda$, security parameter. The domain size of the inputs may be unequal. Let $f_1(x, y)$ be the output generated at the end of the first player P_1 and $f_2(x, y)$ be the output generated at the end of the second player P_2. x and y are chosen according to some joint probability distribution.

The protocol is executed in two phases. In the first phase, the players P_1 and P_2 execute a protocol called Functionality ShareGen. The functionality is parameterized by a real number $\gamma > 0$. The number of rounds of the protocol is $\Omega(\lambda)$. The protocol for share generation is given in Algorithm 1.

In the second phase the parties exchange their shares in the motivation to compute the function. We describe the protocol $\mathbf{\Pi}$ for computing the function in Algorithm 2.

This protocol deals with fail-stop adversary whereas to deal with Byzantine adversary a message-authentication code (MAC) is exploited to resist the players to send false shares. The proof of fairness is same both in fail-stop as well as in Byzantine setting.

Inputs:

1 ShareGen takes the inputs x from P_1 and y from P_2. If one of the received inputs is not in the correct domain, then both the parties are given $\perp$.

Computation:

2 A value $r \in \{1, \cdots\}$ is chosen according to a geometric distribution $\mathcal{G}(\gamma)$ with parameter γ. This represents an iteration (unknown to the parties) in which both the parties get the correct value of $f(x, y)$.

3 The security parameter λ is chosen in such a way so that $\lambda > r$.

4 A set of values $\{a_l\}_{l=0}^{\lambda}$ for P_1 and a set of values $\{b_l\}_{l=0}^{\lambda}$ for P_2 are chosen in such a way that
(i) For $l < r$, $a_l = f_1(x, \hat{y})$ where $\hat{y}$ is the random input variable chosen from the input domain of P_2.
(ii) For $l < r$, $b_l = f_2(\hat{x}, y)$ where $\hat{x}$ is the random input variable chosen from the input domain of P_1.
(iii) For $l \geq r$, $a_l = a_r = f_1(x, y)$.
(iv) For $l \geq r$, $b_l = b_r = f_2(x, y)$.

5 For $l \in \{1, \ldots, \lambda\}$, a_l^1 is chosen randomly from $\{0,1\}^*$, and $a_l^2 = a_l^1 \oplus a_l$ is set.

6 For $l \in \{1, \ldots, \lambda\}$, b_l^1 is chosen randomly from $\{0,1\}^*$, and $b_l^2 = b_l^1 \oplus b_l$ is set.

Output:

7 A list $list_w$ of shares is prepared for each party P_w, where $w \in \{1, 2\}$ such that
(i) P_1 receives the values of $a_1^1, a_2^1, \ldots, a_\lambda^1$ and $b_1^1, b_2^1, \ldots, b_\lambda^1$.
(ii) P_2 receives the values of $a_1^2, a_2^2, \ldots, a_\lambda^2$ and $b_1^2, b_2^2, \ldots, b_\lambda^2$.

Algorithm 1: Functionality ShareGen: Protocol for the share generation

Inputs:

1 P_1 obtains $a_1^1, a_2^1, \ldots, a_\lambda^1$ and $b_1^1, b_2^1, \ldots, b_\lambda^1$.

2 P_2 obtains $a_1^2, a_2^2, \ldots, a_\lambda^2$ and $b_1^2, b_2^2, \ldots, b_\lambda^2$.

Computation:

There are λ number of iterations. In each iteration $l \in \{1, 2, \ldots, \lambda\}$ 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 before P_1 computes any value, P_1 outputs $W_1(x) = f_1(x, \hat{y})$. If P_2 aborts in a round l or the protocol is ended successfully, P_1 outputs the last computed value.

6 If P_1 aborts before P_2 computes any value, P_2 outputs $W_2(y) = f_2(\hat{x}, y)$. If P_1 aborts in a round l or the protocol is ended successfully, P_2 outputs the last computed value.

Algorithm 2: $\mathbf{\Pi}$: Algorithm for computing the output value of the function

In [3], Asharov et al. showed that assuming independent uniform input distribution and a specific set of utilities for each party computing a specific type of functions with complete fairness is impossible following protocol $\mathbf{\Pi}$. They characterized the functions as follows (Definition 4.1 of [3]).

Definition 2 *Let f be a two party function. Let $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ be an input tuple such that $|x_0^0| = |x_0^1| = |x_1^0| = |x_1^1| = n$, and for every $b \in \{0, 1\}$ it holds that*

- $f_1(x_0^0, x_1^b) \neq f_1(x_0^1, x_1^b)$
- $f_2(x_0^b, x_1^0) \neq f_2(x_0^b, x_1^1)$.

From the above definition it is easily seen that a XOR function can be an example of f.

In this direction Groce et al. [10] identified that this impossibility result was achieved due to the utility values assumed by Asharov et al. [3]. We enumerate the utility values assumed in [3] below.

1. Getting correct answer when the opponent outputs a wrong value gives utility 1. In other word, $U_1^{TN} = U_2^{TN} = 1$.
2. Getting incorrect answer while the opponent outputs a correct value gives utility -1 i.e. $U_1^{NT} = U_2^{NT} = -1$.
3. Any other outcomes gives utility 0 i.e. $U_1^{TT} = U_2^{TT} = U_1^{NN} = U_2^{NN} = 0$.

Let us assume that a XOR function is computed following the protocol $\mathbf{\Pi}$. We now investigate whether any player has any incentive to compute the function following the protocol. If a player has no incentive to compute the function, he would abort before the game begins i.e. before the other party will compute any value. Let us assume that P_1 aborts before P_2 computes any value and outputs a random bit according to $W_1(x) = f_1(x, \hat{y})$ over the input x. In case of XOR function he will be correct with probability $\frac{1}{2}$ and will be wrong with probability $\frac{1}{2}$. According to the protocol $\mathbf{\Pi}$, in this case, P_2 will output a random bit according to $W_2(y) = f_2(\hat{x}, y)$, y is the input of P_2. For a XOR function P_2 is also correct

with probability $\frac{1}{2}$ and is incorrect with probability $\frac{1}{2}$. Thus in this case the expected utility of P_1 over the input x should be

$$E(U_1) = \frac{1}{4}(U_1^{TN} + U_1^{TT} + U_1^{NN} + U_1^{NT}).$$

Putting the utility values we get $E(U_1) = 0 = U_1^{TT}$. This is true for any input of P_1. Thus P_1 has no incentive to run the protocol as aborting and cooperating gives him the same utility value. Same argument can be drawn for P_2. Hence the setting is incentive incompatible. However, in [10] Groce et al. modified the utility values as follows.

1. Getting correct answer when the opponent outputs a wrong value gives utility 1. In other word, $U_1^{TN} = U_2^{TN} = 1$.

2. Getting incorrect answer while the opponent outputs a correct value gives utility -1 i.e. $U_1^{NT} = U_2^{NT} = -1$.

3. Getting the correct answer by both the players gives utility $\frac{1}{2}$ i.e. $U_1^{TT} = U_2^{TT} = \frac{1}{2}$

4. Any other outcomes gives utility 0 i.e. $U_1^{NN} = U_2^{NN} = 0$.

Considering these utility values we get that the expected utility of P_1 over any input becomes $\frac{1}{8}$ which is strictly less than $U_1^{TT} = \frac{1}{2}$. The setting is now becomes incentive compatible.

In [10] it is proved that when the setting is incentive compatible, then assuming γ (in [10] it is α) $< \frac{U_1^{TT} - E(U_1)}{U_1^{TN} - E(U_1)}$ the protocol $\mathbf{\Pi}$ achieves fairness for any function. Here, $E(U_1)$ is the maximum expected utility of P_1 when he aborts at the beginning of the game. The maximum is taken over all x that have non-zero probability as input to P_1. It is claimed that the protocol achieves fairness for any function with rational players due to incentive compatibility. By the definition of incentive compatibility the strategy profiles (Cooperation, $W_1(x)$) and (Cooperation, $W_2(y)$) achieve Bayesian Strict Nash equilibrium in ideal world.

However, we identify two classes of functions for which just by modifying the utility values we can not make the setting incentive compatible and hence fairness can not be guaranteed. There exist some inputs x for which $E(U_1(x))$ is always greater than or equal to U_1^{TT} for any value of the utilities and hence those instances are not incentive compatible.

One class deals with the functions which do not have any embedded XOR [8]. The subsequent work of Gordon et al. [9] proved that these type of functions converge to the "greater than function" or more specifically into the millionaire's problem [19]. In millionaire's problem Alice and Bob, two millionaires, want to know who is richer amongst themselves keeping the wealth secret from each other. We describe this function in the next section.

Another class of functions consists of a specific function having an embedded XOR [8]. This function checks whether the two inputs of Alice and Bob are equal or not. We also discuss this function in the next section.

These two classes of functions have great significance in SMC. Since the seminal result of Cleve [5], the community conjectured that two party fair computation is impossible. However, Gordon et al. [8, 9] identified the above two classes of functions for which they

proved that fair two party secure computation is indeed possible. This breaks the decade-old belief about secure two party computation with complete fairness.

4 Incentive Compatible vs. Incentive Incompatible Functions

In this section we recall the mathematical description for incentive compatible settings for a function given in [10]. Then we will show how the functions under consideration do not follow the criterion for any value of the utilities for given guessing strategies and input distribution. We call these functions as incentive incompatible functions. We first describe the greater than function followed by the embedded XOR function.

4.1 Incentive Compatibility

According to [10] incentive compatibility means that there exist two distributions W_1 for P_1 and W_2 for player P_2 such that (Cooperation, $W_1(x)$) and (Cooperation, $W_2(y)$) are Bayesian strict Nash equilibrium in ideal world. More explicitly, let P_1 aborts over a input x at the beginning of the game. We write the expected utility of P_1 in this case as $E(U_1(x))$. If cooperation is a Bayesian strict Nash equilibrium, then $E(U_1(x)) < U_1^{TT}$. This is true for all values of x. In [10] it has been claimed that for incentive compatible setting the maximum expected utility

$$E(U_1) \stackrel{def}{=} \max_x \{E(U_1(x))\} < U_1^{TT},$$

the maximum is taken over all x which have non-zero probability as input to P_1. The maximum expected utility for P_2 has been defined similarly.

We observe that there are some functions which do not satisfy the above condition for any value of the utilities. There exist some inputs x for which $E(U_1(x)) \geq U_1^{TT}$ for all value of utilities assuming the guessing strategies and input distribution as considered by Groce et al. [10]. Thus, there is no possibility so that $E(U_1) < U_1^{TT}$. We call these functions as incentive incompatible functions. This observation helps us to get a clearer view about the structure of incentive incompatible functions and how they differ from incentive compatible functions (functions whose setting can be changed from incentive incompatible to incentive compatible by modifying the utility values).

4.2 Greater than Function

In case of greater than function two players, say P_1 and P_2 posses two secret values i and j respectively, where $1 \leq i, j \leq n$ and n is the domain size of each input. It is assumed that n is polynomial in the security parameter λ. Both the players want to know whether $i > j$ or $i \leq j$. The functionality $f(x_i, y_j)$ is 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 x_i (y_j) is the input value and $f_1(x_i, y_j)$ ($f_2(x_i, y_j)$) is the output

value of the player P_1 (P_2). Here, before share generation, P_1 is sent the ordered list $X = (x_1, x_2, \cdots, x_n)$ and P_2 is sent the ordered list $Y = (y_1, y_2, \cdots, y_n)$. Each party chooses the input from the corresponding list in such a way so that the index of the input matches with the secret value he possesses. This is why P_1 chooses x_i and P_2 chooses y_j. The function is defined as follows [8, 9]. For $w = 1, 2$,

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

We illustrate the above function by the following table.

	y_1	y_2	y_3	y_4	y_5	y_6	y_7
x_1	0	0	0	0	0	0	0
x_2	1	0	0	0	0	0	0
x_3	1	1	0	0	0	0	0
x_4	1	1	1	0	0	0	0
x_5	1	1	1	1	0	0	0
x_6	1	1	1	1	1	0	0
x_7	1	1	1	1	1	1	0

From the above table it is clear that when P_1 chooses x_1 i.e. when his secret value is x_1, he should have no incentive to play the game. In this scenario, he needs no cooperation from P_2 to compute the function as with certainty he knows that the output value must be zero whatever be the input value of P_2. In other words, his output does not depend upon the input values of P_2. In this situation, it is very natural for him to abort the game. In this case if P_1 aborts, P_2 will outputs $W_2(y_j)$ which is equal to $f_2(\hat{x}, y_j)$ according to the protocol $\mathbf{\Pi}$. Thus, the expected utility of P_1 is

$$E(U_1(x_1)) = \Pr(f_2(\hat{x}, y_j) = 0)U_1^{TT} + \Pr(f_2(\hat{x}, y_j) = 1)U_1^{TN}.$$

From the above table it is clear that when $y_j = y_7$, $\Pr(f_2(\hat{x}, y_j) = 1) = 0$. In this case $E(U_1(x_1)) = U_1^{TT}$. In all other cases $E(U_1(x_1)) > U_1^{TT}$. Thus this instance is incentive incompatible. As $E(U_1(x_1)) \geq U_1^{TT}$, there is no possibility that $E(U_1) < U_1^{TT}$. Thus this function is incentive incompatible for any value of utilities under the guessing strategies and input distribution considered by Groce et al. [10].

4.3 Unequal Domain Size

For unequal domain size the above observation remains valid. Let the domain size of P_1 be 7 whereas the domain size of P_2 is 6. In this case the above matrix reduces to the following one.

	y_1	y_2	y_3	y_4	y_5	y_6
x_1	0	0	0	0	0	0
x_2	1	0	0	0	0	0
x_3	1	1	0	0	0	0
x_4	1	1	1	0	0	0
x_5	1	1	1	1	0	0
x_6	1	1	1	1	1	0
x_7	1	1	1	1	1	1

One can easily find that for both x_1 and x_7, P_1 should have no incentive to play the game. This is because when he chooses x_1 or x_7, immediately he comes to known the value of the output. For both the instances it is straight forward to show that according to the protocol $\mathbf{\Pi}$, $E(U_1(x_i))$ is always greater than U_1^{TT} implies $E(U_1) \not< U_1^{TT}$.

4.4 Embedded XOR Function

Now we will discuss the embedded XOR function proposed by Gordon et al. [8].

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 Functionality ShareGen. P_2 also randomly chooses the input from his list and delivers to the ShareGen. For convenience, we here recall the table for f given in [8].

	y_1	y_2
x_1	0	1
x_2	1	0
x_3	1	1

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

$$f(x_i, y_j) = \begin{cases} 1 & \text{if } i \neq j; \\ 0 & \text{if } i = j. \end{cases} \tag{2}$$

Note that for the instance $x_i = x_3$, P_1 has no incentive to play as he knows in certainty that the output should be 1. In this case, according to the guessing strategies $(W_1(x_i), W_2(y_j))$, P_1 always has expected utility $\left[\frac{2}{3}U_1^{TT} + \frac{1}{3}U_1^{TN}\right]$, which is always greater than U_1^{TT} for any value of utilities. Thus, this function is also incentive incompatible for any value of the utilities assuming the guessing strategies and input distribution as in [10].

From the above examples we can define the above functions as follows.

Definition 3 *$f : X \times Y \to \{0, 1\}$ is a two-party function, for which there exists at least one input $x \in X$ such that $f(x, y) = f(x)$, where $|X| = |Y| = n$, n be polynomial in security parameter λ.*

We call these functions as incentive incompatible as for any value of utilities, P_1 has no incentive to compute the functions for some input x even in the ideal world.

Now, from the above analysis we can state the following result.

Theorem 1 *Let $f(x, y)$ be a two-party function defined in 3 and $\mathcal{R}_1 : U_w^{TN} > U_w^{TT} \geq U_w^{NN} \geq U_w^{NT}$ (section 2.6) be the preferences of the players on the utility values, then for any value of the utilities the function behaves as incentive incompatible function under the assumptions that the inputs are chosen according to some joint probability distribution D and the parties follow the guessing strategies $W_1(x)$, $W_2(y)$ specified by the protocol $\mathbf{\Pi}$ in case of abort.*

Proof: Let f be an incentive compatible function for some input distribution D and some utility values. According to the definition of incentive compatibility, the maximum expected utility $E(U_w)$ of a party w who aborts the game, is less than U_w^{TT}. The maximum is taken over all the inputs of the player. Let P_1 aborts the game over the input $x \in X$ for which $f(x, y) = f(x)$. In this case, as the output depends on P_1's input only, P_1 can compute the correct output with probability 1. In this case, according to the protocol $\mathbf{\Pi}$, P_2 will output a bit which depends on $W_2(y)$. If $W_2(y) = f(x, y) = f(x)$, P_2 will be correct, otherwise he will be wrong. Hence the expected utility of P_1 over the input x is

$$\begin{aligned} E(U_1(x)) &= \Pr\Big(W_2(y) = f(x, y)\Big)U_1^{TT} + \Pr\Big(W_2(y) \neq f(x, y)\Big)U_1^{TN} \\ &\geq U_1^{TT} \end{aligned}$$

as from $\mathcal{R}_1 : U_w^{TN} > U_w^{TT} \geq U_w^{NN} \geq U_w^{NT}$, we get $U_w^{TN} > U_w^{TT}$.

This implies that $E(U_1) \not< U_1^{TT}$ which contradicts with the assumption that f is an incentive compatible function for some input distribution D and some utility values.

■

5 Concluding remarks

In [10] Groce et al. claimed that in incentive compatible setting any function can be computed with complete fairness in the context of two party computation when the players are rational. They provided a mathematical description about incentive compatible setting and showed that modifying some utility values incentive incompatible setting for a XOR function can be converted into incentive compatible setting. However, there is no answer to the question that whether modifying the utility values incentive incompatible setting could always be converted into incentive compatible setting for any arbitrary function. To find the answer of the question, we observe that there exist two distinct classes of functions which remain incentive incompatible for any value of the utilities for some guessing strategies and input distribution. One class consists of the functions without any embedded XOR whereas another class has one function with an embedded XOR. These functions had been used to show the first fair two party computation with non-rational players [8, 9]. These observations provide us clearer view about the incentive compatible and incentive incompatible functions.

References

[1] I. Abraham, D. Dolev, R. Gonen, J. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. *Proceedings of the 25th Annual ACM Symposium on Principles of distributed computing, New York, USA*, 53–62, (2006).

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

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

[4] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology* **13**, 1, 143–202, (2000).

[5] 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).

[6] 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).

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

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

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

[10] A. Groce, J. Katz. Fair computation with rational players. *Advances in Cryptology - EUROCRYPT 2012*, 81–98, Springer Berlin Heidelberg, (2012).

[11] 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).

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

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

[14] A. Lysyanskaya, N. Triandopoulos. *Advances in Cryptology - CRYPTO' 06*, 180–197, (2006).

[15] A. Lysyanskaya, A. Segal. IACR Cryptology ePrint Archive : Report 2010/540, (2010).

[16] 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).

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

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

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