Journal of Ambient Intelligence and Humanized Computing (2019) 10:2913–2927
https://doi.org/10.1007/s12652-017-0669-z

ORIGINAL RESEARCH

Rational adversary with flexible utility in secure two-party computation

Yilei Wang[1] · Shuaifeng Zhang[2] · Yi Tang[3] · Qingtang Su[1] · Beijing Chen[4]

Received: 17 October 2017 / Accepted: 23 December 2017 / Published online: 12 January 2018

Abstract
Secure two-party computation evaluates a function among two distributed parties without revealing the parties' inputs except for the function's outputs. Secure two-party computation can be applied into various fields like cloud computing, which is a composition of distribute computing, parallel computing and utility computing etc. Rational secure two-party computation may achieve some desirable properties under two assumptions deriving from STOC 2004. However, the emergence of new computing paradigms like pay-as-you-go model restricts the application of rational protocols. Previous adversaries does not consider payment in secure two-party protocols. Therefore, new type of adversaries should be propose for these new paradigms. In this paper, we address this problem by proposing a new kind of rational adversary, who consider payment in his relaxed utilities. The utilities are based on economic incentives instead of standard assumptions. Furthermore, the new rational adversary is assumed to negotiate with rational parties in protocols. It's similar to "cost corruption" but more flexible. Our new adversary can dynamically negotiate with each rational party in different phases in order to maximize his utilities. To verify the validity of the new adversary, we model a rational secure two-party protocol, which inherits the hybrid framework of STOC 2007. We also prove the security in the presence of the new rational adversary under ideal/real paradigm.

Keywords Rational adversary · Flexible utility · Secure two-party computation

1 Introduction

Rational secure two-party computation allows two rational parties to learn the output of a function without leaking any information. Meanwhile rational parties maximize their utilities. Utility function is a common notion in game theory, which assigns a number for each possible outcome. Utility presents the motivation for rational parties since a higher utility implies preference on the corresponding outcome. Generally, rational parties (Halpern and Teague 2004) (1) prefer the outcome of learning the computation result and (2) others not learning the result. That is, rational parties hope to have an advantage over others. Rational computation in the presence of rational parties may handle difficult tasks in cryptography like fairly information exchanging (Kol and Naor 2008; Gupta et al. 2016) and Byzantine agreement (Groce et al. 2012). Therefore, it's meaningful to delve into the utility functions of rational secure two-party computation. It's similar to deep learning enforcing a specific behaviors, which optimizes algorithms according to the accumulation of experiential data. It is well known that deep learning is a hot topic in machine learning, which

✉ Yilei Wang
wang_yilei2000@126.com

Shuaifeng Zhang
zsfeng2007@163.com

Yi Tang
ytang.bjs@139.com

Qingtang Su
sdytsqt@163.com

Beijing Chen
nbutimage@126.com

[1] School of Information and Electrical Engineering, Ludong University, Yantai, China

[2] Shandong Huayu Aerospace Technology Co.,Ltd., Yantai, China

[3] Department of Mathematics, Guangzhou University, Guangzhou, China

[4] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China

can be applied into various fields (Yu et al. 2017; Ibtihal and Hassan 2017), like biomedical data analysis (Jararweh et al. 2017; Atawneh et al. 2017), neural network (Liang et al. 2015; Leung et al. 2014; Gu et al. 2017) and image processing etc. (Chan et al. 2015; Zhu et al. 2016; Ruijin et al. 2016). Furthermore, it can reinforce learning a specific behaviour (e.g. cooperation) to solve specific problems (Wen et al. 2015; Bin et al. 2015a; Chang et al. 2017b; Gu et al. 2015b; Chang et al. 2017a). In this paper, we try to facilitate rational adversary with relaxed utility definition. Most utility functions inherit from the basic structure of Halpern and Teague (2004), which limit the application scenarios of rational computation. The utility functions are fixed and lack of extensibility to new computation paradigms. So new flexible utility functions should be proposed to fit in with these new requirements. Note that, the non-adversary parties in rational protocols are rational as well. Therefore, it's possible for the adversary to "corrupt" them.

For example, rational adversary is fit for new computation paradigm like pay-as-you-go model (Al-Roomi et al. 2013) to complete computation in cloud (Ren et al. 2015; Li et al. 2014a, b, 2017a, b). Therefore in cloud computing, the preferences on each outcome and utility functions should include payment or cost. Unfortunately, utility functions in existing works only rely on the achievement of security properties like privacy and correctness without considering payment or cost. It's reasonable for an external adversary in cloud to pay for the corruption if he "corrupts" rational parties. It's similar to the work of Garay et al. (2013). The distinction is that rational parties are conditional corrupted after they negotiate with the adversary. That is, corrupted parties at least get payment from the adversary. Note that corrupted parties get nothing when they are corrupted in previous works. Another problem is about the dynamic performance in could computing, where the adversary may corrupt different parties during the computation. Consequently, the utilities for different corruption may be also different, which may incur flexible utility definition for the rational adversary. Therefore, there are three practical problems when we define the rational adversary: (1) preferences on outcome and flexible utility functions including payment; (2) cost corruption when the adversary negotiate; (3) dynamical corruption. To the best of our knowledge, previous definitions on rational adversaries do not cover all three problems. Consequently, new preferences and utilities should be redefined based on these changes.

In this paper we propose a new rational adversary under flexible utilities to address the above three problems in cloud computing. The new rational adversary has flexible utilities when they corrupt different subset of parties. Therefore, the security discussion of two-party computation in the presence of the new adversary is more practical since the assumption on adversaries' ability is more close to the actual behavioural characters. The main contributions of this paper are as follows.

1. We propose a new rational adversary under flexible utilities, who consider corruption cost and dynamically negotiate with parties during the computation.
2. We redefine the views for parties considering history and utilities with respect to the new rational adversary. More specifically, we combine the definition in Groce and Katz (2012) and the bit coin cost (Ruffing et al. 2015; Bentov and Kumaresan 2014; Andrychowicz et al. 2014a; Kumaresan and Bentov 2014; Andrychowicz et al. 2014b).
3. We follow the hybrid framework of Moran et al. (2009), Katz (2007), Gordon et al. (2008), Gordon and Katz (2012), Groce and Katz (2012) prove security under ideal/real paradigm in the new rational adversary. To the best of our knowledge, this is the first time to prove security under such a practical scenario.

1.1 Related works

Halpern and Teague (2004) propose rational secret sharing, which is a specific example of multi-party computation. However, their protocol is non-deterministic and does not fit for two parties. Gordon and Katz (2006) revisit the problem of rational secret sharing in the presence of two rational parties, which is simpler than the work of Halpern and Teague (2004). Maleka et al. (2008a, b) introduce repeated games in rational secret sharing to propose a deterministic protocol. The utility definitions of these works follow the same assumptions of Halpern and Teague (2004). That is, rational parties are assumed to be rational and their utility are fixed. These assumptions on utilities limit the applications of rational protocols.

Asharov and Lindell (2011) relax the standard assumptions on utility functions, which achieves utility independence. While they do not consider corruption cost and dynamical corruption. Micali and Shelat (2009) consider the possibility of learning the wrong secret, which provides a more comprehensive set of utilities. He regards rational secret sharing as a specific kind of mechanism design. In this paper, rational parties participate in the protocol by themselves without considering corruption. Izmalkov et al. (2005) extend rational secure computation to ideal mechanism design, where utilities have more general forms. However, the implementation heavily relies on ballot-box, which is much impractical. Mechanism design theory provides good motivation for flexible utility functions for rational parties. Garay et al. (2013) design a rational protocol between protocol designer and an incentive-driven attacker. Suppose the utility when the attacker achieves privacy and correctness is positive. They also present a natural assumption that corruption of each party brings the attacker negative utility. Note that the adversary can adaptively corrupt parties such that he may control the negative utilities. The attacker's net utility is the sum of the positive and negative one. It obvious that the attacker has incentives to corrupt other parties when the net utility is positive. It reasonably explains the rationality of the attacker. The a fly in the ointment is the neglect of fairness in Garay et al. (2013). The consequently work should focus on finding incentives for rational parties to participate in the protocol. Komatsubara and Manabe (2016) prove the game-theoretic security of bit commitment protocols considering a practical cost model by following the work of Higo et al. (2013). However, their works are specific for commitment protocols and Lack of universality.

Cleve (1986) prove that complete fairness is impossible in secure two-party computation. Generally, fairness can be achieved either by relaxing the notion of fairness like partial fairness (Gordon and Katz 2012) or relax the ability of parties (Ong et al. 2009). Rational protocol belongs to the latter and can achieve fairness in the presence of rational parties. William et al. (2011) achieve fairness in rational secret sharing mixed by rational parties and honest majorities, and later they add malicious parties in their protocol. Their utility definition inherit from Asharov and Lindell (2011) and do not consider corruption cost. Asharov et al. (2011) first give formal definitions of privacy and correctness toward the view of game theory and prove that these two properties can be achieved in rational protocols. Then they give the definition of fairness and prove that the gradual release properties are equal to fair computation. However there is a negative result about fairness. Groce and Katz (2012) revisit this problem and proposed a rational fair protocol for any function. They design new utility definition for rational parties and prove that fairness can be achieved given proper conditions. Their works do not involve corruption cost and dynamical corruption.

The crux of achieving fairness goes back to assigning incentives for rational parties to participate in the protocol. Recently, bit-coins are utilized as a practical incentive for parties to achieve fairness (Andrychowicz et al. 2014a; Bentov and Kumaresan 2014; Kumaresan and Bentov 2014; Andrychowicz et al. 2014b; Jethro 2016; Kiayias et al. 2016). The basic idea is to compensate those who did not learn the output by electronic money like bit-coins (Nakamoto 2009). That is, the adversary corrupts honest parties with cost. He can either learn the result by paying for the honest parties or learn nothing by paying nothing. The cost corruption is similar to the work of Garay et al. (2013).

1.2 Outlines

This paper is organized as follows. Section 2 presents some basic notions such as adaptive security, which will be used in following sections. Furthermore, we give our utility definition considering the cost of rational adversary, new definition of view with respect to history and utility. Section 3 presents our two-party computation protocol in ideal world and hybrid world respectively. In Sect. 4, we use ideal/real paradigm to prove the adaptive security of our hybrid protocol in the presence of rational adversary by constructing an ideal simulator.

2 Background

2.1 Rational two-party protocol and utility definition

Rational protocols can be regarded as games just as mentioned in Alwen et al. (2012). In this paper, we follow the basic notions of transitions between protocols and games (Alwen et al. 2012). Recall that most definitions of utility follow those in game theory such as Prisoner's dilemma game or Stackberg game (Osborne and Rubinstein 1994; Halpern and Teague 2004). These utilities are static once they are defined in the protocols. In fact, utilities maybe change during the protocols. In Groce protocol Groce and Katz (2012), the utilities are defined in the form of matrix (ref. Table 1), where $d_0 > c_0 \geq a_0 \geq b_0$. Note this is common used in game theory when presenting utility for two rational parties.

2.2 Basic framework of a rational two-party protocol

Normally, the security of two-party computation is proved under ideal/real paradigm. There exists a trusted third party in the ideal world protocol, which fulfils all the security properties. However there does not exist such a trusted party in the real world protocol. If the views and outputs of the real world are computationally indistinguishable from the ideal one, the protocol in the real world is secure as that of the ideal one. In this paper, we follow the basic hybrid framework of Moran et al. (2009), Katz (2007), Gordon et al. (2008), Gordon and Katz (2012), Groce and Katz (2012), where the protocol consists of two stages. The first stage is a "pre-processing" step in the presence of a third trusted party and the second stage includes several rounds, where two parties alternatively exchange their messages such that they can finally get the correct output. For example, the first party (say P_1) sends his message to the second party (say P_0) and then P_0 sends his information to P_1. Figure 1 presents the basic idea of the second stage. The hollow circle denotes that it's turn for P_1 to send his message and the solid circle denotes that it's turn for P_0 to send his message. The hollow

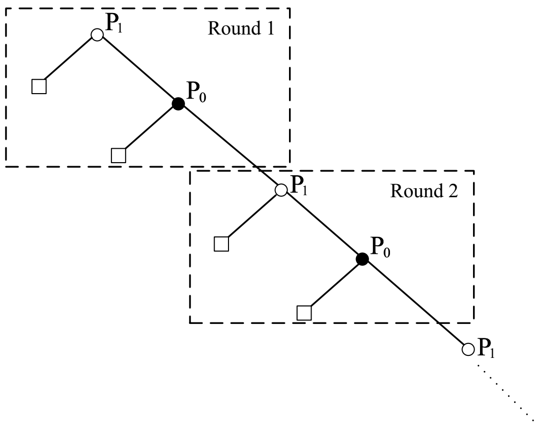

Fig. 1 The flow of the second stage

Table 1 Utility definition of the Groce protocol

	Correct	Incorrect
Correct	(a_0, a_1)	(b_0, c_1)
Incorrect	(c_0, b_1)	(d_0, d_1)

square denotes that the protocol ends and the bigger square with dashed line denotes one round of the second stage.

To prove the security of the protocol, we only need to construct an ideal simulator which can simulate the behaviours of real adversary.

2.3 Views and computational indistinguishability

The basic notions about views are as follows Goldreich (2001, 2009).

- Let x_0, x_1 be inputs for P_0 and P_1 respectively. Let $f(x_0, x_1) = (f_1(x_0, x_1), f_2(x_0, x_1))$ be probabilistic polynomial-time functionality and let π be a two-party protocol to compute functionality $f(\cdot)$. Here we only consider a simple case where $f(x_0, x_1) = f_1(x_0, x_1) = f_2(x_0, x_1)$.
- The view of one party (say x_0) includes the input x_0, random strings r_0^i and received messages m_0^i in the ith round. Therefore, we can denote the view of x_0 as view $^0_\pi$ $(x_0, x_1, n) = (x_0, r_0^i, m_0^i)$, where n is the security parameter. Since the rational adversary adaptively corrupts honest parties, the view of rational adversary is denoted as follows: view $^{adv}_\pi$ $(x_0, x_1, n) = (x_{adv}, r^i_{adv}, m^i_{adv})$. Here x_{adv} is the input of the adversary. In fact it's the input of the corrupted party.
- Denote output$^0_\pi$ (x, y, n) (output$^1_\pi$ (x, y, n)) as the output of honest parties. Denote output$^{adv}_\pi$ (x, y, n) as the output of rational adversary.

A probability ensemble $X = \{X(a, n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \{0, 1\}^*$ and $n \in \mathbb{N}$, where n is the security parameter. Two distribution ensembles $X = \{X(a, n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ and $Y = \{Y(a, n)\}_{a\in\{0,1\}^*;n\in\mathbb{N}}$ are called computationally indistinguishable, which is denoted as $X \overset{c}{\equiv} Y$. It means that for every non-uniform polynomial-time algorithm D, there exists a negligible function $\mu(\cdot)$ such that for $a \in \{0, 1\}^*$ and $n \in \mathbb{N}$,

$$|Pr[D(X(a, n)) = 1] - Pr[D(Y(a, n)) = 1]| \le \mu(n).$$

Definition 1 Let $f(x_0, x_1)$ be a functionality. π securely computers f in the presence of adaptive rational adversaries if there exist probabilistic polynomial-time algorithms $\mathcal{S}$ such that:

$$\mathcal{S}(1^n, x_{adv}, f(x_0, x_1))_{x_0,x_1,n} = \{\mathsf{view}^{adv}_\pi(x_0, x_1, n), \mathsf{output}^{adv}_\pi(x, y, n)\}_{x_0,x_1,n}.$$

3 Some new definitions for rational adversary

3.1 Corruption sequence for rational adversary

In real-world attack, it's hard to fix the subset of corrupted parties beforehand. It is practical to allow rational adversary to dynamically corrupt parties during the protocol. In the second stage of hybrid protocol, two parties alternatively send their messages. The rational adversary may corrupt P_0 or P_1 during the protocol. There are altogether five cases according to the sequence of corruption.

1. Only P_0 is corrupted.
2. Only P_1 is corrupted.
3. First corrupt P_1 and then P_0.
4. Corrupt both P_0 and P_1 at the same time.
5. No one is corrupted.

In the first two cases, only one party is corrupted. The simulator can learn the input of the corrupted party. In the third and fourth cases, two parties are corrupted. Note that the corruption has an order in adaptive corruption setting. In the fourth case, the simulator can learn the inputs and outputs of corrupted parties at the beginning of the simulation. In the third case, the simulator cannot, which makes it much complex to analyze. Note that we do not include the case where P_0 is first corrupted and then P_1 is corrupted since it is identical with the fourth case.

3.2 New utility for rational adversary

As mentioned above, cloud computing is a model to share computing resources like servers, storage and services etc (Li et al. 2018). All these resources need minimal management effort or service provider interaction. Users must pay for it if they want to utilize these resources. Meanwhile the cloud should provide corresponding resources when they are paid. Therefore, it is not free for rational adversary to corrupt parties since they consider their utility at each step. In this paper, the rational adversary pays g bit coins for corrupting each party. Here corrupting means the rational adversary buys the copyright of shares from the corrupted parties. That is, the corrupted parties can not use his shares even if he still holds the shares. We add a session ID in each round of the second stage in the protocol such that one share can be used only once. Therefore, the corrupted parties gets g instead of the result of the protocol. On the other hand, the adversary loses g but gets the result of the protocol. Note that the result of the protocol is either reconstruct the output or not. The original utilities are defined in Table 1 when we did not consider the cost for corrupting. In order to present the utilities when considering corruption cost, we must present new utility definition. It's obvious improper by simply minus cost g in Table 1 since we cannot present utility definition in matrix. Therefore, we present the utility in a tree just like that in the game theory.

Before presenting the utility tree, we first briefly introduce the action set for rational adversary and the flow of the second stage. Let $A = \{\boldsymbol{\Phi}_{P_b}, \bar{\boldsymbol{\Phi}}, follow, abort\}$ is the action set for rational adversary and honest parties. $\boldsymbol{\Phi}_{P_b}$ denotes the action of corrupting party P_b ($b \in \{0, 1\}$) and $\bar{\boldsymbol{\Phi}}$ denotes the action of corrupting no one. When rational adversary adopts $\boldsymbol{\Phi}_{P_b}$, the corrupted party adopts either *abort* or *follow*. In the former case, the corrupted party quits and the protocol ends. In the latter case, the corrupted party follows the protocol. Obviously, the honest parties always adopt *follow*.

Recall that the second stage includes several rounds r, where exists a key round i^*. Parties reconstruct random output before i^* round and correct output after i^* round. In each round i ($1 \le i \le r$), there are two steps.

1. Step 1: P_1 first sends his message to P_0. P_1 may decide whether to send his message.
   (a) Case 1: P_1 is corrupted by the rational adversary. In this case, P_1 will not send message to P_0. Consequently, both parties reconstruct the output by messages received before the ith round. Note that if $i = 1$, then both parties return a random value.
   (b) Case 2: P_1 is not corrupted. In this case, P_1 will send message to P_0. Consequently, the protocol enters into Step 2.
2. Step 2: P_0 sends his message to P_1. It is similar with Step 1.
   (a) Case 1: P_0 is corrupted by the rational adversary. P_0 decides not to send message to P_1. Then P_0 reconstructs the output by using the messages before the ith round. On the other hand, P_1 reconstructs the output by using the messages before the $(i-1)$th round. Note that if $i = 1$, then both parties return a random value.
   (b) Step 2: P_0 is not corrupted. Both parties reconstructs the output by using the messages before the ith round. When we say the messages before the ith round, it includes the message in the ith round.

Note that rational adversary and the honest party may get different utilities in different rounds. Here, we consider the effect of fairness in utility definitions since it's an important property in secure two-party computation. Therefore, we give their utilities in the follow three cases.

The first case $i \in [1, i^* - 1]$.

Under this case, both rational adversary and honest parties reconstructs random values no matter whom rational adversary corrupts. Note that the probability $\frac{1}{|f(x_0,x_1)|}$ that parties guess the correct output is negligible suppose the output range $|f(x_0, x_1)|$ is large enough. Therefore, fairness can be trivially achieved since both adversary and honest parties do not get correct output. Note that rational adversary should pay kg for corrupting parties, where k is the number of corrupted parties and g is the cost for each corruption.

We present the utility definition under this case in Fig. 2, where *Adv* denotes the rational adversary. The shadowed circle denotes that *Adv* decides whether to adopt $\bar{\boldsymbol{\Phi}}$. The dashed hollow circle denotes that the protocol will enter to the next round. The triple denotes the utility of parties, where the first element denotes the utility of *Adv*, the second element denotes the utility of P_1 and the third element denotes the utility of P_0. In this case, all parties cannot get correct output. Therefore, all of them get $d_0 = d_1$ according to the utility matrix in Table 1. If the rational adversary corrupts one party, then the corrupted party will get g as a compensation for sending values to *Adv*.

As mentioned above, there are altogether five cases. We will present the utility definition for each of them.

1. Figure 2a. *Adv* adopts $\bar{\boldsymbol{\Phi}}$ and corrupts no one. Therefore, two honest parties adopt *follow* in each round. Consequently, 0 is the utility for the rational adversary and d_0, d_1 for P_0 and P_1 respectively. Note that at the beginning of the protocol, the rational adversary deposits kg bit coins. Here k is the maximize number of corrupted parties. Each corrupted party will get g after he agrees to be corrupted and sends back proper information to the adversary. Note that we do not use simply bribing the party to prevent the following scenario. One party may

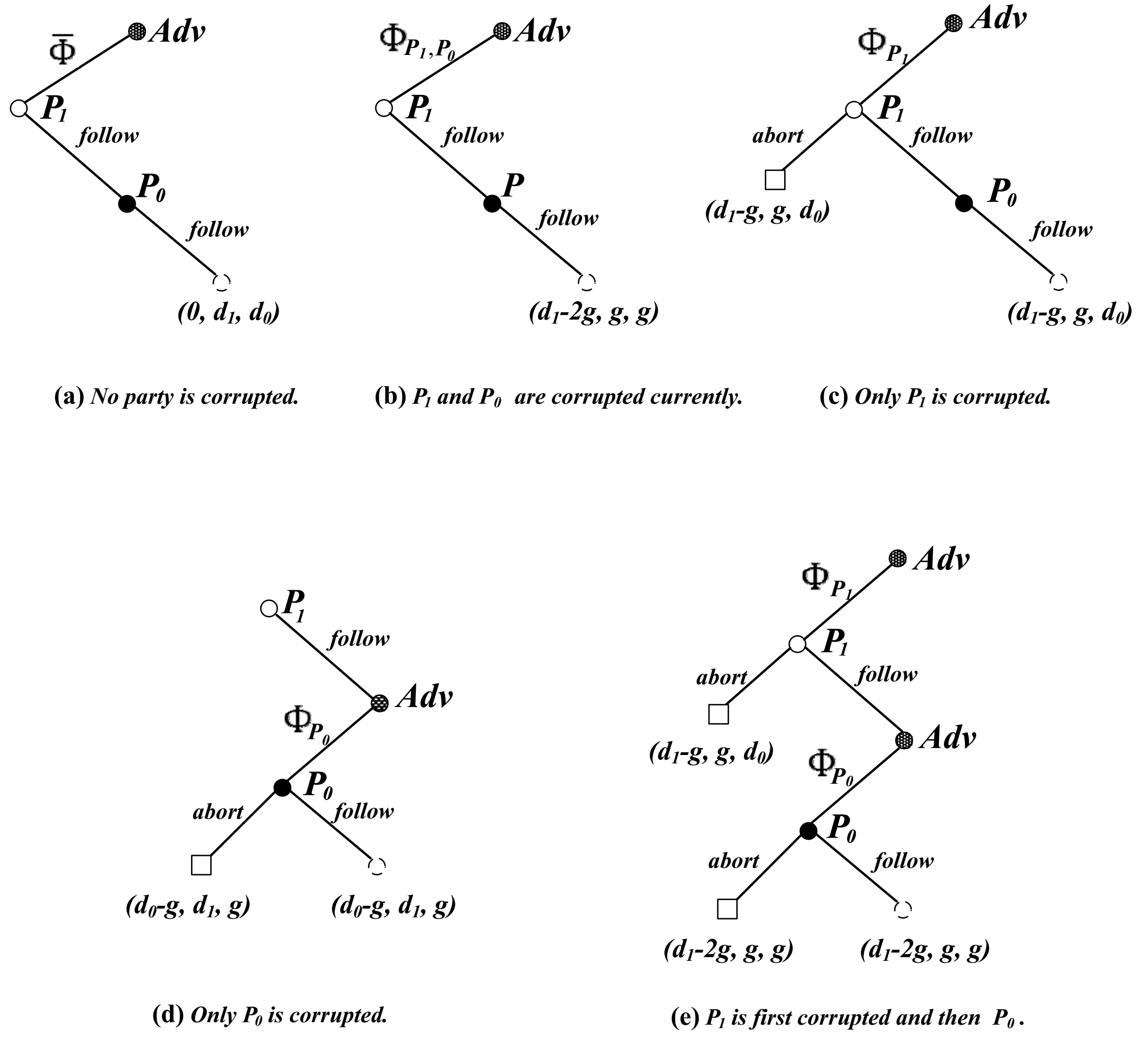

Fig. 2 The utility definition for the case $i \in [1, i^* - 1]$

pretend agree to be corrupted but he may go back on his word after he receives the bribery money. Therefore, we utilize deposits in this paper.

2. Figure 2b. *Adv* adopts Φ_{P_0,P_1} and corrupts P_0, P_1 at the beginning of the protocol. Then two corrupted parties adopt *follow* in each round. Consequently, they will not get any utility since they are corrupted and the utility goes to the rational adversary. *Adv* learns the incorrect output and get utility $d_1 - 2g$ or $d_0 - 2g$. While each honest party gets g. Recall that $2g$ is the corruption cost for two parties. Note that the optimal strategy for them is to "follow" the protocol since "follow" leads to higher utility. Therefore, both parties will coordinate to "follow" when they are corrupted at the same time.
3. Figure 2c. *Adv* adopts Φ_{P_1} and only corrupts P_1. Then *Adv* has two choices.
   (a) If he adopts *abort*, then the protocol ends in Step 1 and both parties (the corrupted P_1 and P_0) learn the incorrect output. *Adv* gets utility $d_1 - g$, P_1 gets g, and P_0 gets utility d_0.
   (b) If he adopts *follow*, then the protocol enters into Step 2. The honest P_0 adopts *follow* in the second step. Consequently, both parties (the corrupted P_1 and P_0) learn the incorrect output. *Adv* gets utility $d_1 - g$, P_1 gets g, and P_0 gets utility d_0. Note that in round $1 \le i \le i^* - 1$, parties always learn incorrect outputs.
4. Figure 2d. The honest P_1 first adopts *follow* and then the rational adversary adopts Φ_{P_0}. As in Fig. 2c. *Adv* has two choices.

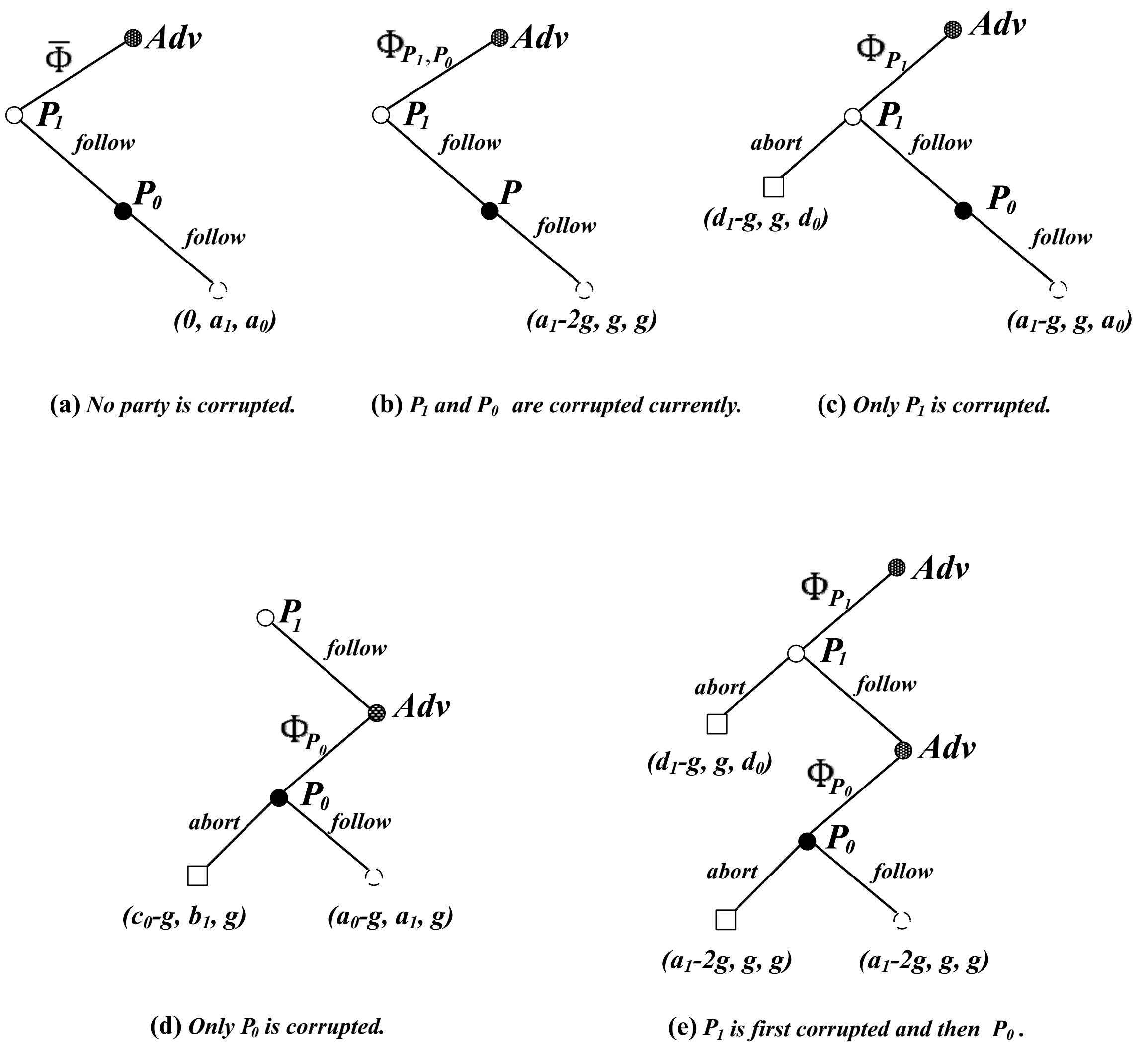

Fig. 3 The utility definition for the case $i = i^*$

(a) If he adopts *abort*, then the protocol ends Step 2 and both parties (P_1 and the corrupted P_0) learn the incorrect output. *Adv* gets utility $d_0 - g$, P_1 gets d_1, and P_0 gets g.
(b) If he adopts *follow*, then the protocol enters into the next round. Consequently, both parties learn the incorrect output. *Adv* gets utility $d_0 - g$, P_1 gets d_1, and P_0 gets g.

5. Figure 2e. *Adv* first adopts Φ_{P_1}. Then *Adv* has two choices.

   (a) If he adopts *abort*, then the protocol ends in the first step and both parties (the corrupted P_1 and P_0) learn the incorrect output. *Adv* gets utility $d_1 - g$, P_1 gets g, and P_0 gets utility d_0.
   (b) If he adopts *follow*, then the protocol enters into the second step. The honest P_0 will adopt *follow* in the second step. However, things are different from Fig. 2c. Here *Adv* then corrupts P_0 at this point. As a rational adversary, the secondly corrupted party P_0 still may adopt *abort* or *follow*.

      i. If he adopts *abort*, the protocol ends in Step 2 and *Adv* learns the incorrect output. *Adv* gets utility $d_1 - 2g$, P_1 and P_0 gets g respectively since they are corrupted.
      ii. If he adopts *follow*, the protocol enters into the next round. Consequently, *Adv* learns the incorrect output gets utility $d_1 - 2g$. At the same time P_1 and P_0 gets g respectively.

The second case $i = i^*$.
This case is much complex. Fairness may be broken if P_0 is corrupted by the rational adversary, who does not send his

message to P_1 in Step 2. The utility of this case is complex as shown in Fig. 3.

1. Figure 3a. *Adv* adopts $\bar{\Phi}$ and corrupts no one. Therefore, two honest parties adopt *follow* in each round. Consequently, 0 is the utility for the rational adversary and a_0, a_1 for P_0 and P_1 respectively.
2. Figure 3b. *Adv* adopts Φ_{P_0,P_1} and corrupts P_0, P_1 currently at the beginning of the protocol. Then two corrupted parties adopt *follow* in each round. Consequently, *Adv* learns the correct output and gets utility $a_1 - 2g$ or $a_0 - 2g$. While the honest parties get g respectively. Recall that $2g$ is the corruption cost for two parties.
3. Figure 3c. *Adv* adopts Φ_{P_1} and only corrupts P_1. Then *Adv* has two choices.

   (a) If he adopts *abort*, then the protocol ends in Step 1 and both parties (the corrupted P_1 and P_0) cannot learn the correct output. *Adv* gets utility $d_1 - g$, P_1 gets g, and P_0 gets utility d_0. In this case, both P_1 and P_0 reconstruct the output of round $i^* - 1$. Recall the output is correct after round i^*.

   (b) If he adopts *follow*, then the protocol enters into Step 2. The honest P_0 adopts *follow* in Step 2. In this case, both parties reconstruct the output of round i^*. Consequently, both parties (the *Adv* and P_0) learn the incorrect output. *Adv* gets utility $a_1 - g$, P_1 gets g, and P_0 gets utility a_0.

4. Figure 3d. The honest P_1 first adopt *follow* and then the rational adversary adopts Φ_{P_0}. *Adv* has two choices.

   (a) If he adopts *abort*, then fairness will be broken. *Adv* (the corrupted P_0) reconstructs the output of round i^* since in Step 1, the honest P_1 sends his message to him. However, in Step 2, *Adv* adopts *Abort* and does not send message to honest P_1. Consequently, P_1 can only reconstruct the output of round $i^* - 1$. Therefore, *Adv* gets utility $c_0 - g$, P_1 gets b_1, and P_0 gets g.

   (b) If he adopts *follow*, then the protocol enters into the next round. Both parties will reconstruct the message of round i^*. Consequently, both parties learn the correct output. *Adv* gets utility $a_0 - g$, P_1 gets a_1, and P_0 gets g.

4. Figure 3e. *Adv* first adopts Φ_{P_1}. Then *Adv* has two choices.

   (a) If he adopts *abort*, then the protocol ends in Step 1 and both parties (the corrupted P_1 and P_0) reconstruct the incorrect output of round $i^* - 1$. *Adv* gets utility $d_1 - g$, P_1 gets g, and P_0 gets utility d_0.

   (b) If he adopts *follow*, then the protocol enters into Step 2. *Adv* corrupts P_0 at this point. As a rational adversary, the secondly corrupted party P_0 adopts *abort* or *follow*. However, it does not affect *Adv* to reconstruct the correct output in round i^*. *Adv* learns the messages of P_0 once he corrupts P_0. Therefore *Adv* gets $a_1 - 2g$ and the corrupted parties get g respectively.

The third case $i \in [i^* + 1, r]$.

Under this case, both rational adversary and honest parties get correct values no matter whom rational adversary corrupts. Therefore, fairness can be trivially achieved since both adversary and honest parties get correct output.

We present the utility definition under this case in Fig. 4. As mentioned above, there are altogether five cases. The definitions are similar to Fig. 2. We only replace d_0 and d_1 with a_0 and a_1 Fig. 4.

3.3 New view definition for rational adversary

Previous works about rational parties/adversaries use traditional view notions. In this paper, we add history and utility into the definition of view. Recall $A = \{\Phi_{P_b}, \bar{\Phi}, follow, abort\}$ is the action set for rational adversary and honest parties. Denote $a_0^i \in A$, $a_1^i \in A$, $a_{adv}^i \in A$ as the actions in the ith round for P_0, P_1 and rational adversary respectively. Denote $h^i = (a_0^i, a_1^i, a_{adv}^i)$ as the action tuple in the ith round and $h^0 = \phi$ as the initial action tuple. Let $H = (h^0, h^1, \ldots)$ denote the history of the protocol. Let $u^i = (u_0^i, u_1^i, u_{adv}^i)$ denote the utility tuple in the *ith* round. Let $U = (u^1, u^2, \ldots)$ denote the utility of the protocol as described in Figs. 2, 3, 4.

Definition 2 The view of one party (say x_0) includes the input x_0, random strings r_0^i, received messages m_0^i, the history tuple h_i and the utility tuple u_i in the ith round. Therefore, we can denote the view of x_0 as view $_\pi^0$ (x_0, x_1, n) $= (x_0, r_0^i, m_0^i, h^i, u^i)$, where n is the security parameter. The view of rational adversary is denoted as follows: view $_\pi^{adv}$ $(x_0, x_1, n) = (x_{adv}, r_{adv}^i, m_{adv}^i, h^i, u^i)$.

4 Ideal and real paradigm

4.1 Ideal world

Just as the model in Moran et al. (2009), Katz (2007), Gordon et al. (2008), Gordon and Katz (2012), Groce and Katz (2012), our ideal model consists of a third trusted party (TTP) and trust ledger for adversary to deposit (Ruffing et al. 2015; Bentov and Kumaresan 2014; Andrychowicz et al. 2014a; Kumaresan and Bentov 2014; Andrychowicz et al. 2014b). The protocol in the ideal world is defined as follows.

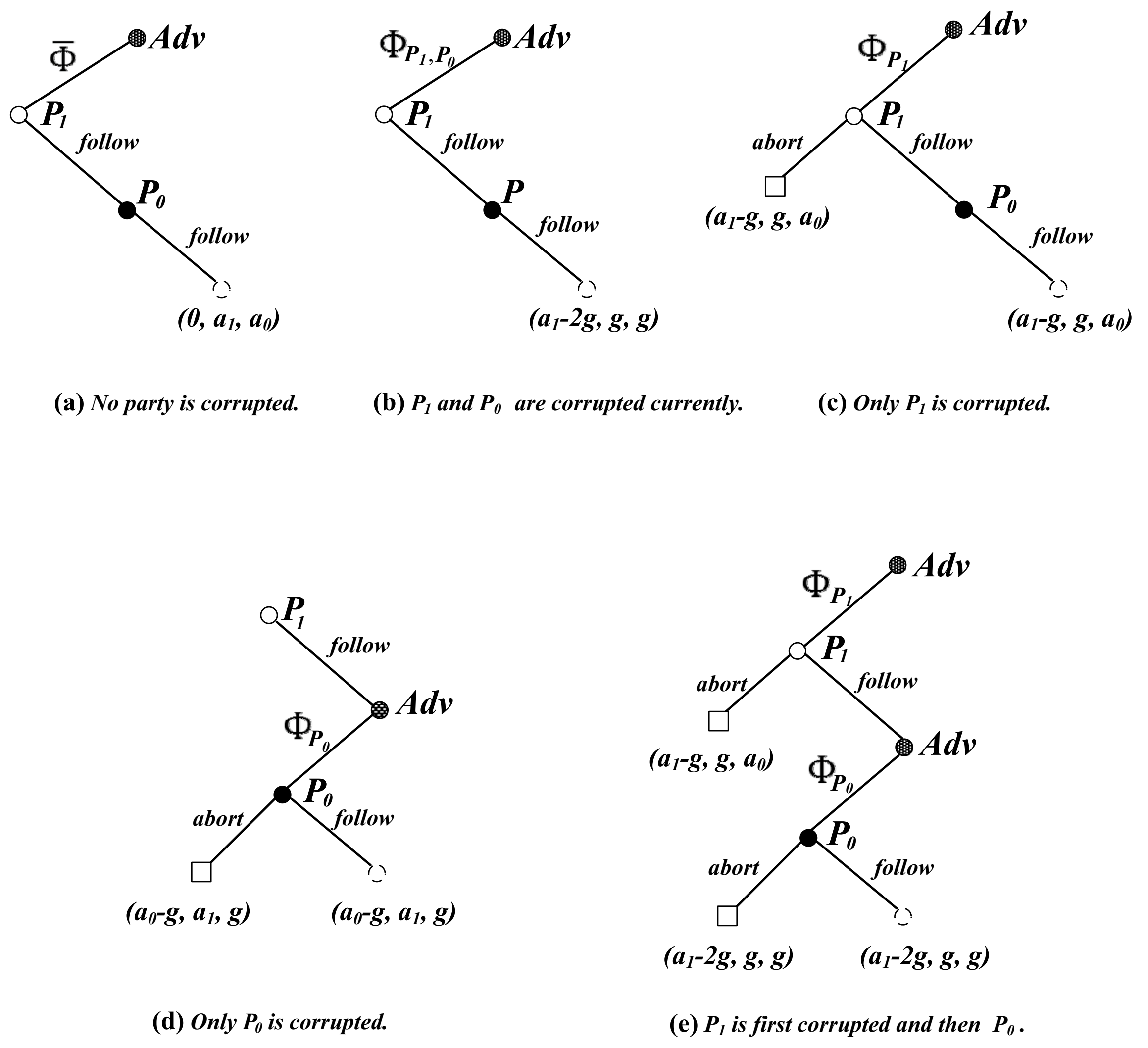

Fig. 4 The utility definition for the case $i \in [i^* + 1, r]$

1. The utility is common knowledge for TTP and parties.
2. Each party chooses his input x_0 (x_1), which are sampled according to a joint probability distribution D over input pairs.
3. Each party sends x'_0 (x'_1), where x'_0 and x'_1 denote the value parties send to the TTP.
4. TTP returns $\perp$ to both parties and the protocol terminates if $x'_0 = \perp$ or $x'_1 = \perp$.
5. Otherwise, TTP computers $f(x'_0, x'_1)$ and returns it to both parties.
6. Honest parties output what they received by TTP. While corrupted parties output what the rational adversary output. Note that rational adversary also learns his utility according to the correctness of its output.

Recall that the rational adversary deposit $2g$ to the ledger. If he corrupts $k \in \{0, 1, 2\}$ parties, then he will get back $(2 - k)g$ bit coins and each corrupted party will get g. The utility whether he gets correct output is obtained according to Table 1. The utility considering the cost of rational adversary is shown in Figs. 2, 3, 4 respectively under different cases. Intuitively, the properties of correctness, privacy and fairness can be all achieved in the ideal world. Then we give our protocol in the real world.

4.2 Real world

The framework of our real world still follows a hybrid protocol (Moran et al. 2009; Katz 2007; Gordon et al. 2008; Gordon and Katz 2012; Groce and Katz 2012), where the first stage is a "pre-processing" step utilizing any protocol for secure two-party computation and the second stage includes $r = \mathcal{O}(n)$ rounds. In each round, there are two steps (ref. Sect. 2.1).

For completeness, we rehearsal the first stage in Groce and Katz (2012).

1. Select i^* according to a geometric distribution p such that parties get incorrect output before round i^* and learn correct output after this round.
2. Assign values r_i^0 and r_i^1 ($i = 1, 2, \dots, r$) to P_0 and P_1 respectively. These values are chosen according to the following rules.
   (a) r_i^0 and r_i^1 are randomly selected in the domain of f when $i < i^*$.
   (b) r_i^0 and r_i^1 are set to be $f(x_0, x_1)$ when $i \geq i^*$.
3. Randomly choose s_i^0 (s_i^1) and t_i^0 (t_i^1) such that $s_i^0 \oplus t_i^0 = r_i^0$ ($s_i^1 \oplus t_i^1 = r_i^1$).

We consider the cost and utility in the second stage, which includes r rounds and in each round, there are two steps.

1. At the beginning of the second stage, rational adversary deposit kg bit coins for k corrupted parties. Note that, rational adversary will returned $(2-k)g$ bit coins if he corrupts k parties and the corrupted party will get g as a compensation. Once the parties are corrupted, he will quit the protocol by getting g. Then the rational adversary and the other party agree on an session ID to exchange the shares.
2. At each round,
   (a) Both parties verify the session ID. If it is verified to be correct, then the start to exchange the shares. Otherwise, they abort. The main task of this step is to prevent reuse of shares. For example, the corrupted parties may reuse the shares after he receives g.
   (b) In the first step, P_1 first sends t_i^0 to P_0.[1] P_0 computes $r_i^0 = t_i^0 \oplus s_i^0$ and the protocol enters into the second step. Otherwise the protocol ends, P_1 does not send t_i^0 to P_0. P_0 and P_1 return r_{i-1}^0 and r_{i-1}^1.
   (c) In the second step, P_0 sends s_i^1 to P_1. P_1 computes $r_i^1 = t_i^1 \oplus s_i^1$. The protocol enters into the next round. Otherwise the protocol ends, P_0 does not send s_i^1 to P_1. P_0 returns r_i^0 and P_1 returns r_{i-1}^1.
3. Both parties leans the latest output if the protocol ends at the last round or end at the ith round.

We present the first and the second stage in Figs. 5 and 6.

[1] There are two cases for P_1 sending t_i^0 to P_0. The first case: P_1 is not corrupted. The second case: P_1 is corrupted by rational adversary but the adversary decides not to deviate from the protocol according to his utility.

Functionality ShareGen

Initialize: The utility definition is common knowledge.
Input: **ShareGen** takes x_0 and x_1 as input for P_0 and P_1 respectively. If either of them is $\perp$, then **ShareGen** return $\perp$ for both parties.
Computation:

1. Choose a key round value i^* according to a geometric distribution with a parameter p.
2. Choose values r_i^0 and r_i^1 according to the following rules.
   (a) If $i < i^*$, choose $r_i^0 \leftarrow W_0(x_0)$ and $r_i^1 \leftarrow W_1(x_1)$. Note that the definitions of W_0 and W_1 are the same to those in Groce and Katz (2012).
   (b) If $i \geq i^*$, let $r_i^0 = r_i^1 = f(x_0, x_1)$.
3. Randomly choose s_i^0 (s_i^1) and t_i^0 (t_i^1) such that $s_i^0 \oplus t_i^0 = r_i^0$ ($s_i^1 \oplus t_i^1 = r_i^1$).

Output: Send s_i^0, s_i^1 to P_0 and t_i^0, t_i^1 to P_1.
Note that all these values are send together with MAC. However, we omit the representation in our protocol just for simplicity and clearness. For the same reason, we omit them in the simulation.

Fig. 5 The first stage of our protocol

The protocol *Π*

Initialize: Parties first get s_i^0, s_i^1 and t_i^0, t_i^1 from the functionality *ShareGen*. rational adversary deposit kg bit coins for k corrupted parties. Both parties agree on a session ID and verify it before each round.
Exchange phase:
There are altogether r rounds and at each round, the protocol proceeds as follows.

1. In the first step,
   (a) P_1 first sends t_i^0 to P_0 if P_1 is not corrupted or corrupted by rational adversary but the adversary decides not to deviate from the protocol according to his utility. P_0 computes $r_i^0 = t_i^0 \oplus s_i^0$ and the protocol enter into the second step.
   (b) Otherwise the protocol ends, P_1 does not send t_i^0 to P_0. P_0 and P_1 return r_{i-1}^0 and r_{i-1}^1.
2. In the second step,
   (a) P_0 sends s_i^1 to P_1 if P_0 is not corrupted or corrupted by rational adversary but the adversary decides not to deviate from the protocol according to his utility. P_1 computes $r_i^1 = t_i^1 \oplus s_i^1$. The protocol enters into the next round.
   (b) Otherwise the protocol ends, P_0 does not send s_i^1 to P_1. P_0 returns r_i^0 and P_1 returns r_{i-1}^1.

Output phase: Parties output the latest value they can reconstruct after the above rounds. For example, P_0 may output r_i^0 or r_{i-1}^0.

Fig. 6 The second stage of our protocol

5 Security and utility in the presence of rational adversary

5.1 The security analysis

In this section, we will prove the security of our hybrid protocol. We use the traditional ideal/real paradigm except for our new view definition. Furthermore, the rational adversary corrupted honest parties in a dynamical way. In addition, we should consider the history and utility when constructing the simulator. Note that in Gordon and Katz (2012), they consider how to construct simulator in the presence of static adversary. In Garay et al. (2013), although dynamic corruption is considered, history and utility do not appear in the simulation construction. In this paper, rational adversary decides to corrupt which parties during the execution of the protocol according to their history and utility. Therefore, utility should be considered in the simulator construction.

Theorem 1 Let Π^{SG} *be the hybrid model, where ShareGen is an functionality. For every non-uniform, polynomial-time rational adversary* $\mathcal{A}^{ral}$ *who adaptively corrupts honest parties, there exist a non-uniform, polynomial-time adversary* $\mathcal{S}^{ral}$ *in the ideal world computing f such that the following equations establish. Note that* $\mathcal{S}^{ral}$ *corrupts the same parties as the rational adversary.*

$$\{IDEAL_{f,\mathcal{S}^{ral}(aux)}(x_0, x_1, n)\}_{x_0,x_1 \leftarrow D, aux \in \{0,1\}^*, H, U}$$
$$\overset{c}{\equiv} \{HYBRID_{\Pi^{SG},\mathcal{A}^{ral}(aux)}(x_0, x_1, n)\}_{x_0,x_1 \leftarrow D, aux \in \{0,1\}^*, H, U}$$

Proof As previous works of simulator construction, we present the simulator $\mathcal{S}^{ral}$ with black-box accessing to $\mathcal{A}^{ral}$. Also similar to Gordon and Katz (2012), we omit the MAC-tags and keys. The simulator is much complex since the rational adversary corrupt parties in a dynamic way. Fortunately, we only discuss two-party computation in this paper. Therefore, there are altogether five cases with respect to the corruption sequence.

1. Corrupt both P_0 and P_1 at the same time. It is trivial to construct a simulator since he learns all the inputs at the beginning of the simulation. Note that rational adversary will not get back his $2g$ deposit.
2. No one is corrupted. No simulator will be constructed in this case.
3. Only P_1 is corrupted.
4. Only P_0 is corrupted.
5. First corrupt P_1 and then P_0.

Consequently, we will stress on the last three cases, which have the same initial process. That is, once the rational adversary $\mathcal{A}^{ral}$ decides to corrupt parties, the simulator $\mathcal{S}^{ral}$ invokes rational adversary the auxiliary input, and the security parameter n, initial history ϕ and utility definition U in Figs. 2, 3, 4. However, the simulator cannot learn the input of the corrupted parties since the rational adversary do not know which parties to corrupt. We inherit the basic idea of Gordon et al. (2008) to construct a simulator.

Only P_1 is corrupted

1. $\mathcal{S}^{ral}$ invokes $\mathcal{A}^{ral}$ on the cost of g bit coins.
2. $\mathcal{S}^{ral}$ invokes $\mathcal{A}^{ral}$ on the input x_1 once $\mathcal{A}^{ral}$ corrupts P_1. The simulator choose x_1' according to distribution D. Here D is defined the same as that of Groce and Katz (2012).
3. $\mathcal{S}^{ral}$ sends x_1 to the functionality *ShareGen* as if the corrupted party does in the real world.
4. $\mathcal{S}^{ral}$ sets $r = p \cdot O(n)$ and uniformly chooses shares t_i^0 and t_i^1, where $i \in [1, 2 \ldots, r]$. $\mathcal{S}^{ral}$ returns the random shares to $\mathcal{A}^{ral}$ as if its output from the computation of *ShareGen*.
5. If the rational adversary sets $x_1 = \perp$ according to utility, then $\mathcal{S}^{ral}$ sends x_1' to the functionality *ShareGen* computing f. $\mathcal{S}^{ral}$ outputs whatever $\mathcal{A}^{ral}$ outputs and the protocol ends. Otherwise, the protocol proceeds as follows.
6. Choose i^* according to geometrical distribution with parameter p.
7. For $i = 1$ to $i^* - 1$

   (a) If $\mathcal{S}^{ral}$ chooses x_0' uniformly from it domain, computes $r_i^1 = f(x_0', x_1)$ and sets $s_i^1 = t_i^1 \oplus r_i^1$. It gives s_i^1 to $\mathcal{A}^{ral}$. $\mathcal{A}^{ral}$ computes his utility according to the history and his output.
   (b) If $\mathcal{A}^{ral}$ decides to take the action *abort*, then $\mathcal{S}^{ral}$ outputs whatever $\mathcal{A}^{ral}$ outputs and the protocol ends.

8. For $i = i^*$ to r

   (a) If $i = i^*$ then $\mathcal{S}^{ral}$ sends x_1 to f and gets $z = f(x_0, x_1)$. Then $\mathcal{S}^{ral}$ sets $s_i^1 = t_i^1 \oplus z$. It gives s_i^1 to $\mathcal{A}^{ral}$. $\mathcal{A}^{ral}$ computes his utility according to the history and his output.
   (b) If $\mathcal{A}^{ral}$ decides to take the action *abort*, then $\mathcal{S}^{ral}$ outputs whatever $\mathcal{A}^{ral}$ outputs and the protocol proceed into next round.

   item If $\mathcal{A}^{ral}$ never aborted (and all r iterations are done), $\mathcal{S}^{ral}$ outputs what $\mathcal{A}^{ral}$ outputs and halts.

The construction of the case where only P_0 is corrupted is similar to the case where only P_1 is corrupted. We omit the description here due to limit of space.

First corrupt P_1 and then P_0

In this case, the simulator can wait until $\mathcal{A}^{ral}$ corrupt P_0 and P_1. Note that, if $\mathcal{A}^{ral}$ first corrupt P_1 and $\mathcal{A}^{ral}$ decides to adopt *abort* in the first step. Then the protocol ends, thus this case is similar to the case where only P_1 is corrupted since $\mathcal{A}^{ral}$ has no chance to corrupt P_0. In this case we discuss the case, where $\mathcal{A}^{ral}$ first corrupts P_1 and $\mathcal{A}^{ral}$ decides not to adopt *abort*. Consequently, $\mathcal{A}^{ral}$ corrupts P_0. Note in this case, $\mathcal{A}^{ral}$ cost $2g$ bit coins.

1. $\mathcal{S}^{ral}$ first invokes $\mathcal{A}^{ral}$ on the cost of g bit coins. Then $\mathcal{S}^{ral}$ invokes $\mathcal{A}^{ral}$ if he has the chance to corrupt P_0 later. This will affect the cost and utility for the rational adversary and simulator.
2. $\mathcal{S}^{ral}$ invokes $\mathcal{A}^{ral}$ on the input x_0 and x_1 until $\mathcal{A}^{ral}$ corrupts P_0 and P_1. The simulator choose x_1' according to distribution D since $\mathcal{A}^{ral}$ may adopt *abort* in the second step. Here D is defined the same as that of Groce and Katz (2012).
3. $\mathcal{S}^{ral}$ sends x_0 and X_1 to the functionality *ShareGen* as if the parties do in the real world.
4. $\mathcal{S}^{ral}$ sets $r = p \cdot O(n)$ and uniformly chooses shares s_i^0, s_i^1 and t_i^0, t_i^1, where $i \in [1, 2 \ldots, r]$. $\mathcal{S}^{ral}$ returns the random shares to $\mathcal{A}^{ral}$ as if its output from the computation of *ShareGen*.
5. Choose i^* according to geometrical distribution with parameter p.
6. For $i = 1$ to $i^* - 1$
   (a) $\mathcal{S}^{ral}$ chooses $x_0'$and x_1' uniformly from it domain, computes $r_i^1 = f(x_0', x_1)$, $r_i^0 = f(x_0, x_1')$ and sets $t_i^0 = s_i^0 \oplus r_i^0$, $s_i^1 = t_i^1 \oplus r_i^1$. It gives t_i^0 and s_i^1 to $\mathcal{A}^{ral}$. $\mathcal{A}^{ral}$ computes his utility according to the history and his output.
   (b) If $\mathcal{A}^{ral}$ decides to take the action *abort*, then $\mathcal{S}^{ral}$ outputs whatever $\mathcal{A}^{ral}$ outputs and the protocol ends.
7. For $i = i^*$ to r
   (a) If $i = i^*$ then $\mathcal{S}^{ral}$ sends x_0 to f and gets $z = f(x_0, x_1)$. Then $\mathcal{S}^{ral}$ sets $t_i^0 = s_i^0 \oplus z$ and $s_i^1 = t_i^1 \oplus z$. It gives t_i^0 and s_i^1 to $\mathcal{A}^{ral}$. $\mathcal{A}^{ral}$ computes his utility according to the history and his output.
   (b) If $\mathcal{A}^{ral}$ decides to take the action *abort*, then $\mathcal{S}^{ral}$ outputs whatever $\mathcal{A}^{ral}$ outputs and the protocol proceed into next round.

   item If $\mathcal{A}^{ral}$ never aborted (and all r iterations are done), $\mathcal{S}^{ral}$ outputs what $\mathcal{A}^{ral}$ outputs and halts.

From the construction of simulator, we can see that it suffices that,

$$\{IDEAL_{f,\mathcal{S}^{ral}(aux)}(x_0, x_1, n)\}_{x_0,x_1 \leftarrow D, aux \in \{0,1\}^*, H, U} \overset{c}{\equiv} \{HYBRID_{\Pi^{SG}, \mathcal{A}^{ral}(aux)}(x_0, x_1, n)\}_{x_0,x_1 \leftarrow D, aux \in \{0,1\}^*, H, U}$$

■

5.2 The utility analysis

To achieve the property of fairness, we must analysis the utility of the rational adversary such that $\mathcal{A}^{ral}$ has no incentives to take *abort*. That is, $\mathcal{A}^{ral}$ is willing to take Φ_{P_b}.

Then we analyze the utility under different corruption cases.

1. When no one is corrupted, then rational adversary will get 0 since he costs 0 and learn nothing. The honest parties will get a_0 and a_1.
2. When two parties all corrupted, then the rational adversary will get $a_0 - 2g$. In this case, the rational adversary learns the output. However, he may pay $2g$ to learn the output. Whilst the honest parties get g.
3. When only P_1 is corrupted.
   (a) In round 1 to i^*, the rational adversary will not learn the output and the honest party P_0 will not learn the output. Therefore, the rational adversary gets utility $d_1 - g$ and P_0 gets utility d_0.
   (b) In round $i^* + 1$ to r, the rational adversary will learn the output and the honest party P_0 will also learn the output. Therefore, the rational adversary gets utility $a_1 - g$ and P_0 gets utility a_0.
4. When only P_0 is corrupted.
   (a) In round 1 to $i^* - 1$, the rational adversary will not learn the output and the honest party P_1 will not learn the output. Therefore, the rational adversary gets utility $d_0 - g$ and P_1 gets utility d_1.
   (b) In round i^*, as rational adversary, he may adopt *abort*. In this case, the rational adversary will learn the output and the honest party P_1 will not learn the output. Therefore, the rational adversary gets utility $b_0 - g$ and P_1 gets the utility c_1.
   (c) In round $i^* + 1$ to r, the rational adversary will learn the output and the honest party P_1 will also

Table 2 The utility of rational adversary under different corruptions

Case	1 to $i^* - 1$	i^*	i^* to r
P_0 is corrupted	$d_0 - g$	$b_0 - g$	$a_0 - g$
P_1 is corrupted	$d_1 - g$	$d_1 - g$	$a_1 - g$
P_1 is corrupted then P_0	$d_0 - g$	$b_0 - 2g$	$a_0 - g$
All is corrupted	$a_0 - 2g$	$a_0 - 2g$	$a_0 - 2g$
No one is corrupted	0	0	0

learn the output. Therefore, the rational adversary gets utility $a_0 - g$ and P_1 gets utility a_1.

5. When P_1 is first corrupted and then P_0 is corrupted.
    (a) In round 1 to $i^* - 1$, the rational adversary will not learn the output and the honest party P_1 will not learn the output. Therefore, the rational adversary gets utility $d_0 - g$ and P_1 gets utility d_1.
        i. In round i^*, if the rational adversary P_1 will adopt *abort*, then rational adversary has no chance to corrupt P_0. Consequently, the protocol ends. In this case, the rational adversary will not learn the output since he does not learn the value in round i^*. Note that the rational adversary learns the value in round $i^* - 1$. Then the utility of rational adversary is $d_0 - g$.
        ii. In round i^*, if the rational adversary P_1 will not adopt *abort*, then the protocol enters into the second step. In this case, the rational adversary corrupts P_0, then the rational adversary will learn the output. Therefore, the rational adversary gets utility $a_0 - 2g$.
    (b) In round $i^* + 1$ to r, the rational adversary will learn the output and the honest party P_1 will also learn the output. Therefore, the rational adversary gets utility $a_0 - g$ and P_1 gets utility a_1 (Table 2).

If all utilities are lower than 0, then rational adversary has no incentives to corrupt any parties. That is:

$$f(x) = \begin{cases} g < d_0; \quad g < d_1; \quad g < \dfrac{a_0}{2}. \\ g < b_0; \quad g < d_1; \quad g < \dfrac{b_0}{2}; \quad g < \dfrac{a_0}{2}. \\ g < a_0; \quad g < a_1; \quad g < \dfrac{a_0}{2}. \end{cases}$$

Recall that $b_0 > a_0 \geq d_0 \geq c_0$. So given $d \leq \frac{a_0}{2}$ and $g \leq min\{d_0, d_1\}$, the rational adversary has no incentives to adopt *abort*. Therefore, fairness is achieved.

6 Conclusions and future works

Cloud computing is a hot topic in information technology field, where security issues bear the brunt in most settings. It combines distributed resources to improve computational efficiency, which becomes the bottleneck for its further development. Rational secure two-party computation, which solves security computation between two parties, provides an effective way to solve these security issues. However, general assumptions on rational adversaries cannot be directly applied in cloud computing settings. We should design new paradigms for rational secure two-party computation since there are new characters in cloud computing due to the introduction of utilities. One task of the paradigms is to design more practical and efficient protocols in cloud computing. In this paper, we consider a two-party computation in the presence of rational adversary. The distinct from previous works lies that we consider the adversary as rational, who has dynamical utilities during the computation. We redefine the utilities for rational adversary by game tree. Then, we redefine the views for rational adversary and try to reflect it in the simulator construction. Finally we prove the security of this protocol in the presence of rational adversary.

The new adversary proposed in this paper only adapts to two-party settings and only simple equilibrium is considered in this paper. The future works include considering the multi-party computation in the presence of rational adversaries. Under this case, there are lots of problems should be solved. For example, a set of parties may collude to maximize their utility. The second problem is to consider adaptive corruption with erase. That is, parties may get partial information of the protocol. The last problem is to consider stronger equilibriums in the new setting.

Acknowledgements This work is partially supported by National Natural Science Foundation of China (Nos. 61502218, 61771231), Shandong Province Higher Educational Science and Technology Program (J14LN20), Natural Science Foundation of Shandong Province (ZR2017MF010, ZR2014FM005), Shandong Province Science and Technology Plan Projects (2015GSF116001), Ph.D. Programs Foundation of Ludong University (Nos. LY2014033, LY2015033), Fujian Provincial Key Laboratory of Network Security and Cryptology Research Fund (Fujian Normal University) (No. 15004). Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) and Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET). Yi Tang is supported by Guangzhou scholars project (No. 1201561613).

References

Al-Roomi M, Al-Ebrahim S, Buqrais S, Ahmad I (2013) Cloud computing pricing models: a survey. Int J Grid Distrib Comput 6(5):93–106

Alwen J, Jonathan K, Ueli M, Vassilis Z (2012) Collusion-preserving computation. In: Advances in Cryptology–CRYPTO 2012, Springer, New York, pp 124–143

Andrychowicz M, Dziembowski S, Malinowski D, Mazurek Ł (2014a) Fair two-party computations via bitcoin deposits. In: Financial cryptography and data security. Springer, New York, pp 105–121

Andrychowicz M, Dziembowski S, Malinowski D, Mazurek L (2014b) Secure multiparty computations on bitcoin. In: 2014 IEEE symposium on security and privacy. IEEE, pp 443–458

Asharov G, Lindell Y (2011) Utility dependence in correct and fair rational secret sharing. J Cryptol 24(1):157–202

Asharov G, Canetti R, Hazay C (2011) Towards a game theoretic view of secure computation. Advances in cryptology-EUROCRYPT 2011. Springer, New York, pp 426–445

Atawneh S, Almomani A, Hussein AB, Putra S, Brij G (2017) Secure and imperceptible digital image steganographic algorithm based on diamond encoding in dwt domain. Multimed Tools Appl 76(18):18451–18472

Beekman JG (2016) A denial of service attack against fair computations using bitcoin deposits. Inf Process Lett 116(2):144–146

Bentov I, Kumaresan R (2014) How to use bitcoin to design fair protocols. In: Advances in cryptology-CRYPTO 2014. Springer, New York, pp 421–439

Chan T-H, Jia K, Gao S, Lu J, Zeng Z, Ma YP (2015) A simple deep learning baseline for image classification? IEEE Trans Image Process 24(12):5017–5032

Chang X, Ma Z, Lin M, Yang Y, Hauptmann A (2017a) Feature interaction augmented sparse learning for fast kinect motion detection. IEEE Trans Image Process 26(8):3911–3920

Chang X, Yu Y-L, Yang Y, Xing EP (2017b) Semantic pooling for complex event analysis in untrimmed videos. IEEE Trans Pattern Anal Mach Intell 39(8):1617–1632

Cleve R(1986) Limits on the security of coin flips when half the processors are faulty. In: Proceedings of the 18th annual ACM symposium on theory of computing, ACM, pp 364–369

Garay J, Katz J, Maurer U, Tackmann B, Zikas V (2013) Rational protocol design: cryptography against incentive-driven adversaries. In: Foundations of computer science (FOCS), 2013 IEEE 54th annual symposium on, IEEE, pp 648–657

Goldreich O (2001) Foundations of cryptography: volume 1, basic tools. Cambridge University Press, Cambridge

Goldreich O (2009) Foundations of cryptography: volume 2, basic applications. Cambridge University Press, Cambridge

Gordon DS, Carmit H, Katz J, Lindell Y (2008) Complete fairness in secure two-party computation. In: Proceedings of the 40th annual ACM symposium on theory of computing, ACM, pp 413–422

Gordon SD, Katz J (2006) Rational secret sharing, revisited. In: Security and cryptography for networks. Springer, New York, pp 229–241

Gordon DS, Katz J (2012) Partial fairness in secure two-party computation. J Cryptol 25(1):14–40

Groce A, Katz J (2012) Fair computation with rational players. In: Annual international conference on the theory and applications of cryptographic techniques. Springer, pp 81–98

Groce A, Katz J, Thiruvengadam A, Zikas V (2012) Byzantine agreement with a rational adversary. Automata, languages, and programming. Springer, New York, pp 561–572

Gu B, Sheng VS, Tay KY, Romano W, Li S (2015a) Incremental support vector learning for ordinal regression. IEEE Trans Neural Netw Learn Syst 26(7):1403–1416

Gu B, Sheng VS, Wang Z, Ho D, Osman S, Li S (2015b) Incremental learning for ν-support vector regression. Neural Netw 67:140–150

Gu B, Sun X, Sheng VS (2017) Structural minimax probability machine. IEEE Trans Neural Netw Learn Syst 28(7):1646

Gupta B, Agrawal DP, Yamaguchi S (2016) Handbook of research on modern cryptographic solutions for computer and cyber security. IGI Glob

Halpern J, Teague V (2004) Rational secret sharing and multiparty computation: extended abstract. In STOC 2004: Proceedings of the 36th annum ACM symposium on theory of computing, New York, ACM, pp 623–632

Higo H, Tanaka K, Yasunaga K (2013) Game-theoretic security for bit commitment. International workshop on security. Springer, New York, pp 303–318

Ibtihal M, Hassan N (2017) Homomorphic encryption as a service for outsourced images in mobile cloud computing environment. Int J Cloud Appl Comput (IJCAC) 7(2):27–40

Izmalkov S, Micali S, Lepinski M (2005) Rational secure computation and ideal mechanism design. In: Foundations of computer science, 2005. FOCS 2005. 46th Annual IEEE symposium on, IEEE, pp 585–594

Jararweh Y, Al-Ayyoub M, Fakirah M, Alawneh L, Gupta BB (2017) Improving the performance of the Needleman–Wunsch algorithm using parallelization and vectorization techniques. Multimed Tool Appl (3):1–17

Katz J (2007) On achieving the best of both worlds in secure multiparty computation. In Proceedings of the 39th annual ACM symposium on theory of computing, ACM, pp 11–20

Kiayias A, Zhou H-S, Zikas V (2016) Fair and robust multi-party computation using a global transaction ledger. In: Annual international conference on the theory and applications of cryptographic techniques, Springer, pp 705–734

Kol G, Naor M (2008) Cryptography and game theory: designing protocols for exchanging information. In: Fifth theory of cryptography. Springer, New York, pp 320–339

Komatsubara Tsuyoshi, Manabe Yoshifumi (2016) Game-theoretic security of commitment protocols under a realistic cost model. In Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on, pages 776–783. IEEE

Kumaresan R, Bentov I (2014) How to use bitcoin to incentivize correct computations. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, ACM, pp 30–41

Leung MKK, Xiong YH, Lee LJ, Frey BJ (2014) Deep learning of the tissue-regulated splicing code. Bioinformatics 30(12):i121–i129

Li J, Chen X, Li M, Lee PPC, Li J, Lou W (2014a) Secure deduplication with efficient and reliable convergent key management. Parallel Distrib Syst IEEE Trans 25(6):1615–1625

Li J, Huang X, Li J, Chen X, Xiang Y (2014b) Securely outsourcing attribute-based encryption with checkability. Parallel Distrib Syst IEEE Trans 25(8):2201–2210

Li J, Zhang Y, Chen X, Xiang Y (2018) Secure attribute-based data sharing for resource-limited users in cloud computing. Comput Secur 72:1–12

Li P, Li J, Huang Z, Gao C-Z, Chen W-B, Chen K (2017a) Privacy-preserving outsourced classification in cloud computing. Cluster Comput pp 1–10

Li P, Li J, Huang Z, Li T, Gao C-Z, Yiu S-M, Chen K (2017b) Multi-key privacy-preserving deep learning in cloud computing. Future Gen Comput Syst 74(C):76–85

Liang M, Li Z, Chen T, Zeng J (2015) Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach. IEEE/ACM Trans Comput Biol Bioinf (TCBB) 12(4):928–937

Maleka S, Shareef A, Rangan CP (2008a) The deterministic protocol for rational secret sharing. Parallel and distributed processing. In: IEEE international symposium on parallel and distributed processing, pp 1–7

Maleka S, Shareef A, Rangan CP (2008b) Rational secret sharing with repeated games. Information security practice and experience. Springer, New York, pp 334–346

Micali S, Shelat A (2009) Purely rational secret sharing. Theory of cryptography. Springer, New York, pp 54–71

Moran T, Naor M, Segev G (2009) An optimally fair coin toss. In: Theory of cryptography. Springer, New York, pp 1–18

Moses Jr WK, Rangan CP (2011) Rational secret sharing over an asynchronous broadcast channel with information theoretic security. arXiv preprint arXiv:1112.4033

Nakamoto S (2009) Bitcoin: a peer-to-peer electronic cash system. Counsulted

Ong SJ, Parkes DC, Rosen A, Vadhan SP (2009) Fairness with an honest minority and a rational majority. In: Theory of cryptography. Springer, pp 36–53

Osborne MJ, Rubinstein A (1994) A course in game theory. MIT Press, Cambridge

Ren Y, Shen J, Wang J, Han J, Lee S (2015) Mutual verifiable provable data auditing in public cloud storage. J Int Technol 16(2):317–323

Ruffing T, Kate A, Schröder D (2015) Liar, liar, coins on fire!: penalizing equivocation by loss of bitcoins. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, ACM, pp 219–230

Ruijin Z, Tan Y, Zhang Q, Fei W, Zheng J, Yuan XUE (2016) Determining image base of firmware files for arm devices. IEICE Trans Inf Syst 99(2):351–359

Xuezhi W, Ling S, Yu X, Wei F (2015) A rapid learning algorithm for vehicle classification. Inf Sci 295:395–406

Yu C, Li J, Li X, Ren X, Gupta BB (2017) Four-image encryption scheme based on quaternion fresnel transform, chaos and computer generated hologram. Multimed Tool Appl pp 1–24

Zhu R, Tan Y, Zhang Q, Li Y, Zheng J (2016) Determining image base of firmware for arm devices by matching literal pools. Digital Invest 16:19–28