IEICE TRANS. INF. & SYST., VOL.E99–D, NO.7 JULY 2016

PAPER

Reputation-Based Collusion Detection with Majority of Colluders

Junbeom HUR[†], **Mengxue GUO**[††], **Younsoo PARK**[††], ***Nonmembers***, **Chan-Gun LEE**[††], ***and*** **Ho-Hyun PARK**[††a)], ***Members***

SUMMARY The reputation-based majority-voting approach is a promising solution for detecting malicious workers in a cloud system. However, this approach has a drawback in that it can detect malicious workers only when the number of colluders make up no more than half of all workers. In this paper, we simulate the behavior of a reputation-based method and mathematically analyze its accuracy. Through the analysis, we observe that, regardless of the number of colluders and their collusion probability, if the reputation value of a group is significantly different from those of other groups, it is a completely honest group. Based on the analysis result, we propose a new method for distinguishing honest workers from colluders even when the colluders make up the majority group. The proposed method constructs groups based on their reputations. A group with the significantly highest or lowest reputation value is considered a completely honest group. Otherwise, honest workers are mixed together with colluders in a group. The proposed method accurately identifies honest workers even in a mixed group by comparing each voting result one by one. The results of a security analysis and an experiment show that our method can identify honest workers much more accurately than a traditional reputation-based approach with little additional computational overhead.

key words: *cloud computing, collusion detection, majority voting, reputation*

1. Introduction

In cloud computing, a client distributes computational tasks to multiple workers in a cloud server that are expected to perform their tasks and return the results [1], [2]. Because the workers may not return the true results and the client may be unable to control the workers, it may be unclear whether the returned result is actually true [3], which raises the problem of cloud system integrity [4]–[6]. Occasionally, malicious workers can communicate with each other efficiently, and collectively decide to return the same false results. Such attackers are called colluders [7], [8].

Several methods for colluder detection have been proposed [9]. Quiz-based methods use tasks with verifiable results [7], called 'quizzes', which are embedded into the tasks. When the task server receives the results from the workers, it can confirm that the results of the real tasks are true if all of the quizzes are valid. A quiz method is resilient to collusions, and has an advantage in that the quiz results can be verified [7], [10]. However, there is no method for automatically generating quizzes effectively, which prevents the use of this technique for large-scale projects.

A spot-checking method is a collusion-resistant sabotage-tolerance mechanism [11], which estimates the frequency of malicious attempts that each worker intentionally falsifies the results of assigned jobs, and dynamically eliminates false results from the system. However, this model is not very effective against collusion attacks. Y. Ding et al. proposed a trusted worker scheduling method to detect collusive attackers and assure the integrity of data processing in MapReduce framework [12]. However, the reducers should be fully trusted. Thus, if they collude, there is no way to detect it.

In addition to the above methods, majority-voting methods [13] have been used, such as correlation- and reputation-based methods. Correlation-based methods use the results of votes to partition honest workers from colluders [14]. If the voting results of some of the workers are the same, they will be partitioned into the same group. In this way, a correlation-based method counts how often each pair of workers are together in the same group, and how often they are in opposite groups. It then uses these counts to estimate a distinctive feature, called 'correlation'. If the correlation among some workers is very high, the workers will be grouped together. However, this method does not perform very well in the case of conditional colluders* [14], [15]. In other words, when colluders do not collude often, they are not identifiable and the workers cannot be grouped.

In addition to correlation-based methods, reputation-based methods are commonly used to evaluate the reliability of workers [16]. The workers are partitioned into several groups based on the value of their reputation. The group with the highest reputation value is considered as an honest group. GridEigenTrust [17] combines trust-computation techniques with the EigenTrust reputation rating system [18] in grid computing. Lee et al. proposed a simplified group detection method for detecting colluders, but their method is restricted to colluders forming a group [19]. Bendahmane et al. applied a reputation model to the MapReduce framework for big data processing [20]. H.-C. Tsai et al. proposed a threshold-adaptive reputation scheme for mobile ad hoc networks [21]. E. Abdullah and S. Fujita proposed

Manuscript received August 12, 2015.
Manuscript revised February 28, 2016.
Manuscript publicized April 7, 2016.

[†]The author is with Korea University, Seoul, 02841, Korea.
[††]The authos are with Chung-Ang University, Seoul, 06974, Korea.
a) E-mail: hohyun@cau.ac.kr

DOI: 10.1587/transinf.2015EDP7318

*There are two types of colluders: Unconditional colluders always try to collude to return false results whereas conditional colluders collude only if they know that majority of workers actually collude in a vote [14].

a reputation-based colluder detection scheme for peer-to-peer content delivery networks [22]. The reputation-based methods have been known to be the most successful among majority-voting methods.

However, reputation-based methods work properly only when the number of colluders is no more than one-half of all workers. We perform a simulation and mathematical analysis to figure out the problems of traditional reputation-based methods. In the simulation, we set the proportion of colluders among all workers (P_1) and the probability that a colluder actually participate in the collusion (P_2) as input parameters. The simulation result shows that regardless of P_1 and P_2, if the reputation value of a group is significantly different, i.e., much higher or much lower, from those of other groups, the group is an honest group. Here, the reputation value of a worker is defined as the ratio of the number of correct results to the number of total votes attempted by the worker. The formal definition of the reputation value will be presented by Eq. (1) in Sect. 2.1. Even in the case that the number of colluders is much more than that of honest workers, the group with the lowest reputation value is observed as an honest group. Although this observation contradicts the assumptions of traditional reputation systems, it sometimes happens in the case of majority of colluders. The only case in which an honest group cannot be found is when honest workers are mixed with colluders in the same reputation group.

We analyze the reputation-based method mathematically. We derive a formula to calculate the probability that honest workers are mixed with colluders in a group for all cases of P_1 and P_2. The mathematical analysis almost coincides with simulation results. Based on our rigorous analysis results, we propose a new approach to distinguish honest workers from colluders as follows:

Even though there is no knowledge about P_1 and P_2, just perform the existing reputation-based method. If there is a group whose reputation is significantly higher or lower than those of the other groups, this group is the completely honest group and the search stops. Otherwise, honest workers may be mixed with colluders in the same group and the checking of all groups continues until all honest workers are found.

The remainder of this paper is organized as follows. In Sect. 2, we review a reputation-based method and conduct a simple experiment to measure its grouping accuracy. In Sect. 3, we analyze the grouping characteristics mathematically and propose our scheme based on the analysis result, and in Sect. 4, evaluate its accuracy and level of performance. Finally, in Sect. 5, we provide concluding remarks and a direction for future work.

2. Review of Reputation-Based Collusion Detection

A client may assign the same task to multiple workers in a cloud. Some of the computing results may be right and others may be wrong. Using majority voting, if the voting results of some of the workers are the same, they will be partitioned into the same group. Based on the voting results, all of the workers will be divided into one majority and the other minority groups [14].

However, the accuracy cannot be high for only a single vote. A number of redundant votes are needed to obtain voting reliability. A reputation-based method was therefore proposed. This method is based on the assumption that the number of honest workers is more than the number of colluders.

2.1 Existing Reputation-Based Method

If a client has a number of tasks that need to be computed, it stores them in a queue. For a reputation-based method, workers compute the task chosen from a task queue and then vote. Because there is one task per vote, the number of tasks is equal to the number of votes. The reputation of the i^{th} worker is defined as:

$$R_i = \frac{m_i}{M}, \tag{1}$$

where m_i denotes the number of correct results generated by the i^{th} worker and M is the total number of votes attempted by that worker. In a traditional reputation system, a 'correct' result is the majority answer [23].

In the next step, we partition the workers into multiple groups according to the value of their reputation. The workers in each group have the same reputation value. The group that has the maximum reputation value becomes the honest group [20]. We refer to these traditional reputation methods [20], [23] as the existing method for simplicity. The following example shows how the existing reputation-based method is carried out:

Example 1) Consider workers w_i for $1 \le i \le 7$, and tasks T_j for $1 \le j \le 4$. Assume that $w_1 - w_4$ are honest workers and $w_5 - w_7$ are colluders.

The number of honest workers in Table 1 is more than the number of colluders. If a worker w_j's output for task T_i is true, the value for row T_i and column w_j in Table 1 is set to 1. Otherwise, it is set to 0.

While $w_1 - w_4$ naturally return true results because they are honest workers, w_5 always acts as a colluder by returning false results for all tasks. w_6 behaves as a colluder by returning false results for T_1, T_2 and T_4 whereas it pretends to be an honest worker by returning a true result for T_3. w_7 pretends to be an honest worker for T_1 and T_4 while it colludes for T_2 and T_3.

According to the reputation-based method, $w_1 - w_4$ will

Table 1 An example of an existing reputation-based method

	w_1	w_2	w_3	w_4	w_5	w_6	w_7
T_1	1	1	1	1	0	0	1
T_2	1	1	1	1	0	0	0
T_3	1	1	1	1	0	1	0
T_4	1	1	1	1	0	0	1
R_i	1	1	1	1	0	1/4	2/4

Table 2 Drawbacks of an existing reputation-based method

	w_1	w_2	w_3	w_4	w_5	w_6	w_7
T_1	1	1	0	0	1	1	1
T_2	1	1	0	0	1	1	1
T_3	0	0	0	0	1	1	1
T_4	0	0	0	0	1	1	1
R_i	1	1	2/4	2/4	2/4	2/4	2/4

be partitioned into a group because they have the same reputation value, i.e., one, whereas w_5, w_6, and w_7 will be partitioned separately based on their reputation values (0, 1/4 and 2/4, respectively). In addition, $w_1 - w_4$ are then chosen as honest workers to tell the truth because they compose the group with the highest reputation.

2.2 Grouping Problem

A reputation-based method can efficiently detect both colluders and non-colluders, and guarantee high accuracy with an acceptable amount of overhead [20]. However, this method does have certain limitations. If the number of colluders is more than half of all workers, it cannot discriminate honest workers from colluders because some colluders will be partitioned with honest workers, as shown in the following example.

Example 2) Consider a case of seven workers (w_i) and four tasks (T_i) to vote on. Unlike Example 1, we assume that $w_1 - w_4$ are colluders and that $w_5 - w_7$ are honest workers. We also set the true output to 1 and a false output to zero.

In Table 2, the number of colluders is more than the number of honest workers, and the colluders do not always collude. $w_5 - w_7$ naturally return the true result because they are honest workers. Meanwhile, w_3 and w_4 always act as colluders by returning false results for all tasks. Both w_1 and w_2 pretend to be honest workers for T_1 and T_2 whereas both return false results for T_3 and T_4 by colluding with w_3 and w_4.

For T_1 and T_2, an output of 1 is regarded as correct. On the contrary, for T_3 and T_4, an output of zero is regarded as correct because it forms the majority answer (Four zeros are more than three ones).

According to an existing reputation-based method, colluders w_3 and w_4 will be grouped with actual honest workers $w_5 - w_7$ because they all have the same reputation value of 2/4. Meanwhile, w_1 and w_2 will be chosen as honest workers because their reputation value of 1 is the highest. The group $w_3 - w_7$ including actual honest workers is regarded as a colluders' group because it does not have the highest reputation value. Therefore, it is undesirable to choose the group that has the highest reputation in this case. A new method for identifying honest workers is therefore required.

2.3 Grouping Accuracy

Example 2 shows that, using an existing reputation-based method, some colluders can be chosen as honest workers when there are more colluders than honest workers. This error is caused by incorrect grouping, i.e., w_3 and w_4 were improperly grouped with $w_5 - w_7$. Therefore, before proposing a new method to identify honest workers, it is necessary to evaluate the accuracy of grouping for an existing reputation-based method. In this paper, we assume that honest workers always return true results while colluders behave probabilistically. The colluders may return false results with some probability but not always.

To facilitate the evaluation, we introduce two parameters: P_1, which is the proportion of colluders among all workers, and P_2, which is the probability that a colluder will actually collude. With $1 - P_2$, a colluder may not collude, and act like an honest worker to avoid being identified as a colluder. We assume that all of the colluders have the same probability of P_2. To measure the grouping accuracy, we utilize the failure rate. Some relevant definitions to facilitate this are given below.

Definition 1 (Mixed group) *If there are both honest workers and colluders in a group, the group is called a mixed group.*

The group of $w_3 - w_7$ in Example 2 of Sect. 2.2 is a mixed group.

Definition 2 (Grouping Failure) *After grouping using a reputation-based method, if a mixed group exists, we can say the grouping has failed.*

Definition 3 (Failure Rate) *If grouping is performed K times and a grouping failure occurs f times, the ratio of the number of grouping failures to the number of groupings, i.e., f/K, is called the failure rate.*

An experiment was conducted for a set of $N = 100$ workers and $M = 100$ tasks, where $K = 100$. We set the ranges of P_1 and P_2 to $0.1 - 0.9$, with an interval of 0.05. To facilitate the evaluation, we set the following tasks. We give honest workers input V_{input} randomly from domain [0, 1], and output y is produced as the input received by a worker. This process can be expressed as follows.

$$y = V_{input}$$

We also give colluders an input randomly from domain [0, 1], and at the same time, give them a value of P_2. Each colluder chooses t randomly from domain [0, 1]. If t is in [0, P_2], the worker will generate an improper result denoted by V_{error}. Otherwise, it will generate a true output just like an honest worker. This process can be expressed as follows.

$t \leftarrow$ randomly select from $[0, 1]$
$if\ t \in [0, P_2]$
    $y = V_{error}$
$else$

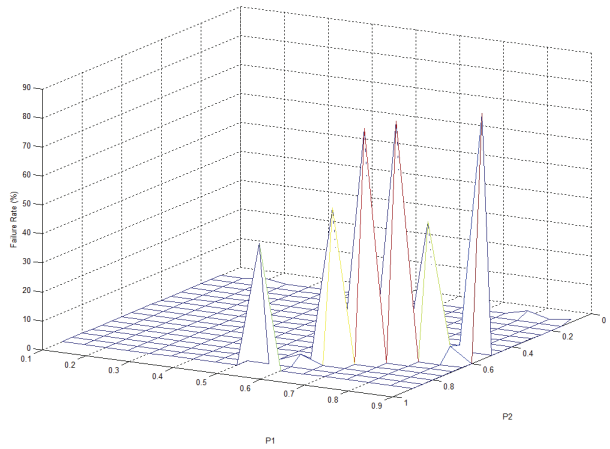


Fig. 1 Failure rate of an existing reputation-based method

$$y = V_{input}$$

We regard the failure rate as a measure of the grouping accuracy.

Figure 1 shows grouping accuracy of an existing reputation-based method using the failure rate. For $P_1 < 0.5$, no matter how P_2 changes, the failure rate is almost zero because honest workers form the majority group and the probability that a colluder will be grouped with honest workers is very low. The following example is provided as a further explanation.

Example 3) In our experiment, when $P_1 < 0.5$, for a colluder to be grouped with honest workers, the 100 voting results must be the same for both. When P_2 has the minimum value, i.e., $P_2 = 0.1$, the possibility that this will occur is at maximum because a colluder almost always produces correct results. In this case, the probability that a colluder will be grouped with honest workers is $(1-0.1)^{100} = 2.66\times10^{-5}$, which is considerably low.

However, when $P_1 > 0.5$, the accuracy of the grouping is indeterminate and changes with the value of P_1 and P_2. In this case, the output of honest workers may be regarded as incorrect. Either honest workers or colluders become the majority for each vote. After 100 votes, the reputation of some colluders may be the same as that of honest workers, and therefore these colluders will be grouped with honest workers, as shown in Example 2. We will analyze why this phenomenon occurs, in Sects. 3.1 and 3.2. Therefore, to solve this problem, a novel approach is needed to determine honest workers with a higher accuracy.

3. Our Method

We observed in Sect. 2 that the existing reputation system has a grouping failure problem when the number of colluders is more than the number of honest workers. In this section, we analyze in detail why such a grouping failure occurs, and suggest an algorithm to solve the problem.

In Sect. 3.1, we observe closely through a more complex example what the reputation values of all groups are when a grouping success/failure occurs. The observation result states that if the number of colluders is less than a half of all workers, namely $P_1 < 0.5$, the existing reputation system always performs successful grouping and the reputation of the honest group is always one. This is a natural phenomenon in a majority voting system [20].

However, if the number of colluders is more than a half of all workers ($P_1 > 0.5$), grouping failures occurs according to the actual collusion probability (P_2) of colluders. If both P_1 and P_2 are very high, the grouping also succeeds. Ironically in this case, the reputation value of the honest group shows lowest. In the situation that most workers are colluders, if they actually collude, false results form a majority group and are regarded as correct. On the other hand, honest workers are considered to be telling a lie.

Next in Sect. 3.2, we mathematically analyze the cases that grouping succeeds and/or fails. The analysis result states that if there is a big reputation difference between groups, grouping succeeds. Otherwise, the probability of a grouping failure is high. This analysis result plays importantly because P_1 and P_2 are not known in a real situation.

While the existing method always considers the highest reputation group as the honest group, our method combines the observation of Sect. 3.1 and the mathematical analysis of Sect. 3.2, and draws the following conclusion: If there is a big reputation difference between groups, a completely honest group necessarily exists and the reputation value of the honest group is highest or lowest.

In Sect. 3.3, when a grouping failure occurs, i.e., honest workers are mixed with colluders within a group, a method to find only honest workers is introduced. By consolidating these analysis and observation, we propose Algorithm 1 in Sect. 3.4.

3.1 Observation of Grouping

Let us take a closer look at the cases of successful grouping and failed grouping through the following example:

Example 4) Assume that the number of workers $N = 50$, and that the number of tasks $M = 50$. The worker ID is used to identify the workers. Honest workers are set in front of all other workers; an honest worker's ID is $(1 - P_1) * N$. Therefore, when $P_1 = 0.7$, an honest worker's IDs becomes $1 - 15$, and when $P_1 = 0.9$, the ID becomes $1 - 5$. In addition, P_2 was set to 0.5 and 0.7. The reputations for these four combinations of P_1 and P_2 were generated, as shown in Table 3, by conducting a reputation-based method.

When both P_1 and P_2 are 0.7, the existing method will falsely recognize only the worker 19 as an honest worker because it has the highest reputation value of 0.68. The actual honest workers ($1 - 15$) are considered as colluders because their reputation value (0.50) is not highest. Some colluders (23, 38 and 39) are grouped with honest workers because

Table 3 Reputation results for a few cases of P_1 and P_2

	$P_1 = 0.7$		$P_1 = 0.9$	
	Reputation	Worker ID	Reputation	Worker ID
$P_2 = 0.7$	0.42	44	0.02	1-5
	0.44	16, 43	0.54	36
	0.46	45	0.60	22, 39, 50
	0.48	25	0.64	15, 18, 19
	0.50	1-15, 23, 38, 39	0.66	7, 9, 12, 20, 21, 26, 28
	0.52	26, 30, 37	0.68	11, 30, 31, 33, 42, 48
	0.54	32, 36, 47, 50	0.70	6, 8, 13, 23, 29, 43
	0.56	17, 20, 40, 42, 46	0.72	16, 25, 32, 40, 46, 49
	0.58	18, 21, 27, 34	0.74	14, 35, 45
	0.60	24, 29, 31, 33, 48	0.76	24, 47
	0.62	22, 35	0.78	17, 27, 34, 38
	0.66	28, 41, 49	0.80	10, 37, 41, 44
	0.68	19		
$P_2 = 0.5$	0.36	37, 48	0.34	37
	0.42	18, 39, 50	0.36	8
	0.44	27, 45	0.46	29, 50
	0.46	16, 25, 26, 32, 42	0.48	15, 45
	0.48	29, 41, 46	0.50	9, 11, 17, 24, 47, 49
	0.50	30, 33, 34	0.52	13, 16, 18, 33, 34, 43
	0.52	19, 20, 23, 24, 35, 38, 47	0.54	7, 30, 31, 44
	0.54	21	0.56	6, 19, 27, 42
	0.56	17, 22, 40, 44	0.58	35
	0.58	43, 49	0.60	10, 20, 21, 22, 32, 36, 39, 48
	0.60	31	0.62	23, 38, 40, 41, 46
	0.62	28, 36	0.64	1-5, 12, 14, 26
	1	1-15	0.66	28
			0.68	25

they all have the same reputation, i.e., 0.50. The difference in the reputation value between each adjacent group is not more than 0.05.

For $P_1 = 0.7$ and $P_2 = 0.5$, the honest workers are well grouped and identified because they have the highest reputation value of 1. There are no mixed groups because no colluders are grouped with honest workers. In addition, the difference between the honest group and the group with the second-highest reputation (0.60) is as high as 0.4.

When $P_1 = 0.9$ and $P_2 = 0.7$, the existing method will mistake the workers 10, 37, 41 and 44 for honest workers because they have the highest reputation value of 0.80. The actual honest workers $(1-5)$ are considered as colluders because their reputation value (0.02) is not highest. There are no mixed groups because no colluders are grouped with honest workers having reputations of 0.02. Even in this case, the honest group has the lowest reputation because the majority of workers are colluders and more than half of them actually collude. The difference between it and the group with the second-lowest reputation (0.54) is significant at 0.52.

In the case of $P_1 = 0.9$ and $P_2 = 0.5$, the existing method will falsely recognize only the worker 25 as an honest worker because it has the highest reputation value of 0.68. The actual honest workers $(1-5)$ are considered as colluders because their reputation value (0.64) is not highest. Some colluders (12, 14 and 26) are grouped with honest workers because they all have the same reputation of 0.64. The difference in the reputation between each adjacent group is not more than 0.05.

In an existing reputation-based method, the group with the highest reputation value is always selected as the honest group. However, in Table 3, this is correct for only one ($P_1 = 0.7$ and $P_2 = 0.5$) of the four cases. Thus, this method should be reconsidered, especially when the number of colluders is more than half of all workers.

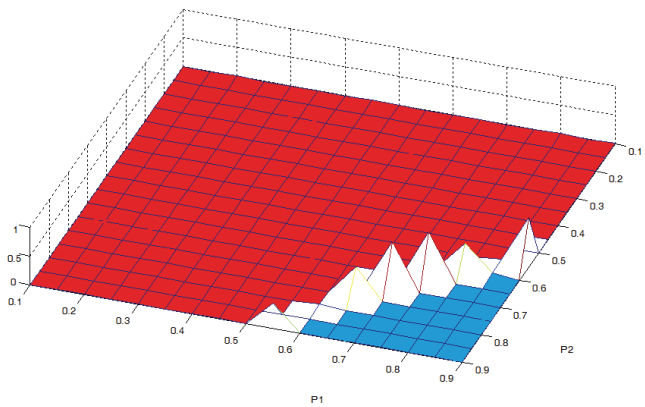


Fig. 2 Aerial view of Fig. 1

Through Table 3, we can see a particular phenomenon: After grouping based on the reputation values, if the reputation value of the honest group is either the highest or lowest, and the difference in the reputation between an honest group and the other groups is significant, no mixed group is generated. If the reputation value of honest workers is neither the highest nor the lowest, and the differences in the reputation among all groups are insignificant, a mixed group is generated. Figure 2 is a redrawing of Fig. 1 that explains this

phenomenon.

In Fig. 2, when P_1 and P_2 are in the red area, the reputation of the honest group is the highest, and is much higher than those of the other collusion groups. When P_1 and P_2 are in the blue area, the reputation of the honest group is the lowest, and is much lower than those of the other collusion groups. In both cases, no mixed groups are generated. However, if the reputation of the honest group is neither too high nor too low, the difference in the reputation among all groups is insignificant. Therefore, colluders can be easily grouped with honest workers. This case is represented by the white mountain-shaped area, which has a nonzero failure rate in Fig. 1. Furthermore, in the blue area of Fig. 2, even though the grouping is successful, honest workers will be regarded as colluders by an existing reputation-based method because they have the lowest reputation value.

Considering the above phenomenon, we construct a sketch of our mechanism as follows: Under the premise of not knowing P_1 or P_2, after using the existing reputation-based method, if there is a group whose reputation is the highest or lowest and that has a significant difference from the reputations of the other groups, we can conclude that no mixed group has been generated and that this group is the honest group.

Otherwise, a mixed group may have been generated, and thus, it is necessary to check all groups starting from the group with the largest size until all honest workers are found. During this process, we denote a 'significant difference' as the threshold, τ_1. In the next section, we first show some probability formulas, and using these formulas, we then determine the threshold, τ_1.

3.2 Theoretical Analysis

Figures 1 and 2 show that when the number of colluders is larger than that of honest workers, after M votes, some colluders may be grouped with honest workers. We calculate the probability of such an occurrence in this section.

After a vote, the probability that a false result will become a majority answer is denoted by P_{b-many}. This may happen only when $P_1 > 0.5$. Thus, we have the following.

$$P_{b-many} = \binom{N \cdot P_1}{N \cdot 0.5} \cdot P_2^{N \cdot 0.5} \cdot (1 - P_2)^{N \cdot (P_1 - 0.5)} + \cdots + \binom{N \cdot P_1}{N \cdot P_1} \cdot P_2^{N \cdot P_1} = \sum_{i=N \cdot 0.5}^{N \cdot P_1} \binom{N \cdot P_1}{i} P_2^i (1 - P_2)^{NP_1 - i} \tag{2}$$

After a vote, the probability that the true result will become a majority answer is denoted by P_{g-many}, which gives us

$$P_{g-many} = 1 - P_{b-many} \tag{3}$$

After a vote, the probability that a colluder will generate a majority answer is denoted by Po_{bad}. We thus have

$$Po_{bad} = (1 - P_2) \times P_{g-many} + P_2 \times P_{b-many} \tag{4}$$

After a vote, the probability that an honest worker will generate a majority answer is denoted by Po_{good}. Because honest workers never collude, the probability that an honest worker will collude is zero. Thus, we have

$$Po_{good} = 1 \times P_{g-many} + 0 \times P_{b-many} = P_{g-many} \tag{5}$$

After M votes, the probability that a colluder will generate majority answers i times is denoted by $P_{bad}(i)$, giving us

$$P_{bad}(i) = \binom{M}{i} \cdot Po_{bad}^i \cdot (1 - Po_{bad})^{M-i} \tag{6}$$

After M votes, the probability that an honest worker will generate majority answers i times is denoted by $P_{good}(i)$. We therefore have the following:

$$P_{good}(i) = \binom{M}{i} \cdot Po_{good}^i \cdot (1 - Po_{good})^{M-i} \tag{7}$$

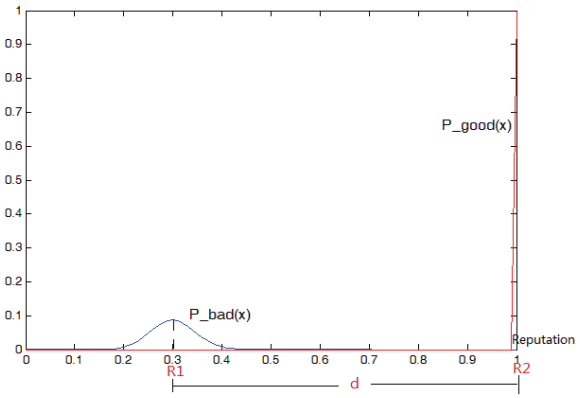
According to the definition of a reputation, $P_{bad}(i)$ and $P_{good}(i)$ can be regarded as the probabilities that a colluder and an honest worker will have a reputation of i/M, respectively. Let $P_{bad}(x)$ and $P_{good}(x)$ be the probabilities that a colluder and an honest worker will have a reputation of x, respectively. Then, $P_{bad}(i)$ and $P_{good}(i)$ can be rewritten as $P_{bad}(x)$ and $P_{good}(x)$, respectively. If the values of P_1 and P_2 are given, we can obtain the probabilities of the reputations, as shown in Fig. 3, for certain values of P_1 and P_2.

The blue curve represents $P_{bad}(x)$, and the red curve represents $P_{good}(x)$. Let $d = |R2 - R1|$ be the absolute value of difference in reputation between the honest group and the collusion groups, where $R1$ and $R2$ are the average reputation values of colluders and honest workers, respectively. As P_1 and P_2 change, d also changes. In the cases shown in Figs. 3 (a) and 3 (d), it is impossible to generate a mixed group because the value of d is sufficiently large. However, in Figs. 3 (b) and 3 (c), generating a mixed group is possible.

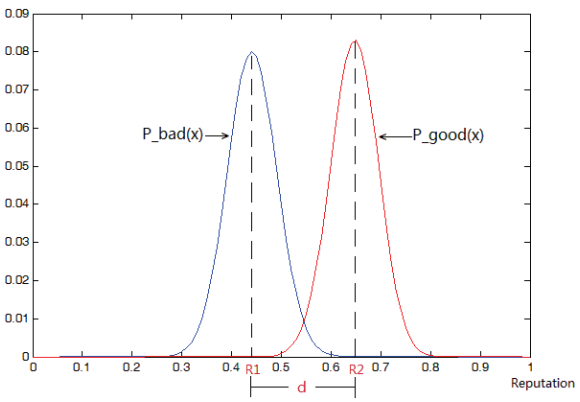
As shown in Fig. 3 (c), in particular, the two curves almost overlap, and d is very small. From these results, we can see that no matter how high or low an honest worker's reputation is, the greater d is, the more unlikely it is that a colluder will be grouped with honest workers.

For a colluder and an honest worker to belong to the same group, they should have the same reputation value. Since an event in which a colluder has a reputation of x, and an event in which an honest worker has a reputation of x, are mutually independent, the probability of both having the same reputation of x is $P_{good}(x) \cdot P_{bad}(x)$. Because all honest workers always have the same reputation, the probability of a colluder and an honest worker belonging to the same group is the same as a colluder and all honest workers belonging to the same group. Therefore, as shown in Fig. 4, when the reputation = x, the probability that a colluder is grouped with honest workers is

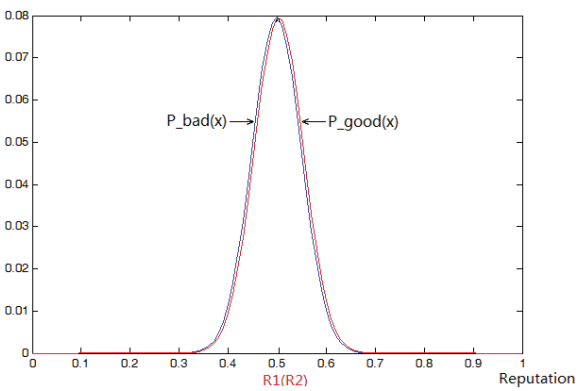
$$f_0(x) = P_{good}(x) \cdot P_{bad}(x) \tag{8}$$



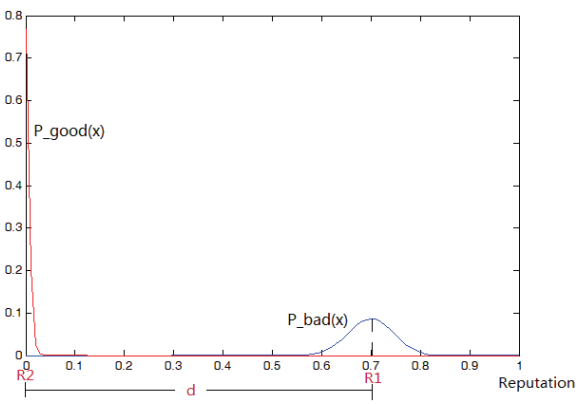
(a) P_1=0.4 and P_2 = 0.7



(b) P_1=0.7 and P_2 = 0.7



(c) P_1=0.72 and P_2 = 0.7



(d) P_1=0.9 and P_2 = 0.7

Fig. 3 Values of $P_{bad}(x)$ and $P_{good}(x)$

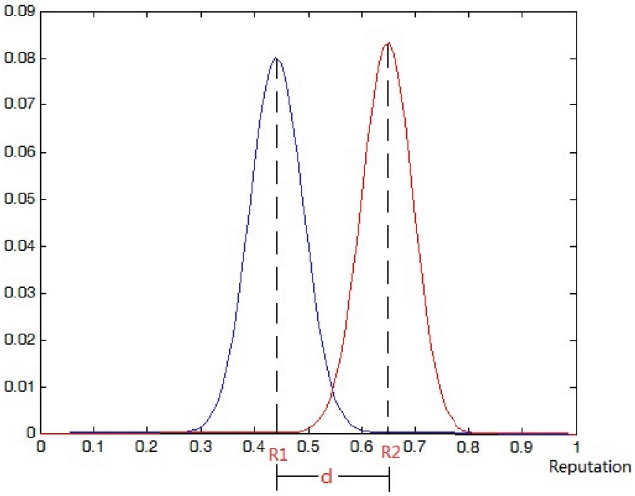


Fig. 4 Example of f_0 when $P_1 = 0.7$ and $P_2 = 0.7$

For all reputation values in the range of 0 to 1, the probability that a colluder will be grouped with honest workers can be denoted by f_0 as follows:

$$f_0 = \int_0^1 P_{good}(x) \cdot P_{bad}(x)\, dx \tag{9}$$

where f_0 is a case in which only one colluder is grouped with honest workers, and there are $P_1 \cdot N$ colluders in total. Therefore, the probability that colluders will be grouped with honest workers is denoted by F as follows:

$$F = \sum_{i=1}^{P_1 \cdot N} \binom{P_1 \cdot N}{i} \cdot f_0^i \cdot (1 - f_0)^{(P_1 \cdot N - i)} \tag{10}$$

According to Definition 2, F is the probability of a grouping failure. We use the above equations to generate graphs for the probability of a grouping failure and the difference in reputations between the honest and collusion groups.

3.2.1 Determination of Threshold τ_1

An evaluation of Eq. (10) was performed for a set of N = 100 workers and M = 100 tasks, where K = 100. We set the range of P_1 at 0.01 − 0.99, which means that there are one to 99 colluders per 100 workers. The range of P_2 is 0−1.

Figure 5 shows an aerial view of the grouping failure rate obtained using Eq. (10). The shape of this figure is very similar to that of Fig. 1. This shows that our analysis coincides with experimental results of Sect. 2.3. As the figure shows, when $P_2 < 0.1$, the failure rate is quite high because, after 100 votes, colluders rarely collude for each vote and mostly act as honest workers. Therefore, colluders can be easily grouped with honest workers, which leads to the generation of a mixed group. Such colluders are very difficult to detect.

The area of the failure rate, < 0.1, in Fig. 5 is almost the same as the area of the difference in reputation, > 0.2,

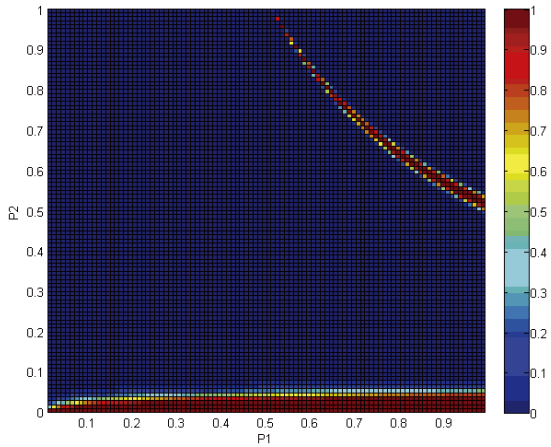


Fig. 5 Aerial view of grouping failure probability

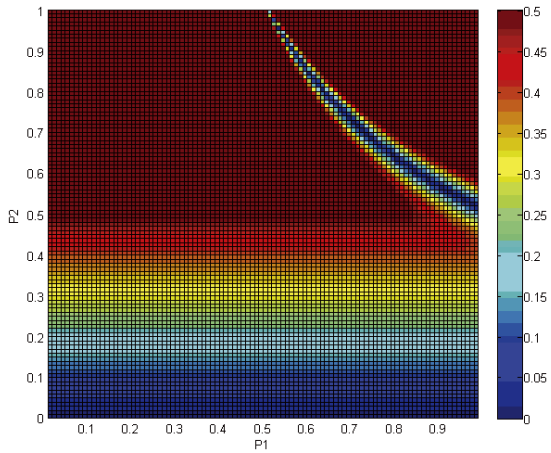


Fig. 6 Aerial view of the reputation difference between the honest group and collusion groups

in Fig. 6. Therefore, we set threshold $\tau_1 = 0.2$ for $M = 100$ and $N = 100$ in our experiment described in Sect. 4.

Thus far, the theoretical background for the observation in Sect. 3.1, that is, if there is a big reputation difference between groups, there must be not a mixed group but a completely honest group, and threshold τ_1 have been provided. However, if a mixed group is generated, we should check all of the groups from the group whose size is largest until all honest workers are found. In the next section, we show how to determine honest workers even when a mixed group exists.

3.3 Identifying the Honest Workers within the Mixed Group

We propose a method to identify honest workers when a mixed group exists. In reputation systems, the computation results of all workers for all tasks are usually stored to calculate their reputations. We assume that this history of computation results is maintained in a table, as shown in Fig. 7, called a computation result table. We construct a computation result table for each group.

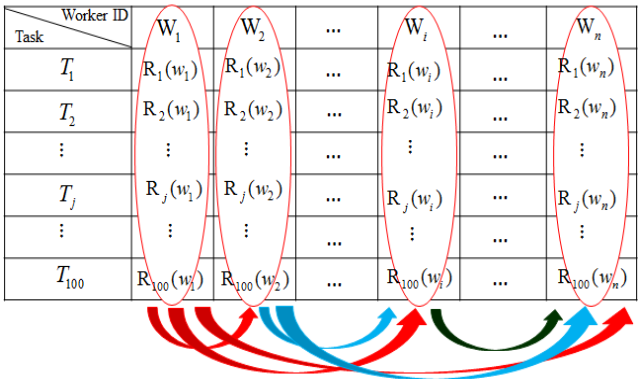


Fig. 7 Computation result table and column-wise comparison

When a grouping fails, the most direct way to identify honest workers is to examine the computation results of all workers for M tasks one by one. Looking at Table 3 in Sect. 3.1 carefully, the group whose size is the largest is most likely to contain honest workers. Therefore, we first check the computation results of workers in the largest group. As shown in Fig. 7, assume there are n workers in this group. It is necessary to make a column-wise comparison of the computation result table as indicated using the arrows in Fig. 7. Here, $R_j(w_i)$ represents the computation result of worker w_i for task T_j. The rationale behind the column-wise comparison is that the column values of honest workers are always the same.

We introduce another threshold, τ_2, as follows: If the number of workers whose column values are all the same is more than threshold τ_2, the workers in this group are honest workers and the check is stopped. Otherwise, the computation result tables of the other groups are continuously checked in this manner until all honest workers are found.

To verify the accuracy of our method, we show that the probability that colluders will be chosen over honest workers is very low. First, let us consider that only one group is generated. Naturally, this group is a mixed group. For colluders to be chosen, the number of colluders whose column values are the same should be more than threshold, τ_2.

For a single task, the probability that the computation results of τ_2 colluders will be the same is denoted by P_{same_0}. This case will happen when either τ_2 colluders collude together or do not collude together. Thus, we have

$$P_{same\_0} = P_2^{\tau_2} + (1 - P_2)^{\tau_2} \tag{11}$$

For M tasks, the probability that the column values of τ_2 colluders will be the same is denoted as P_{same}:

$$P_{same} = [P_{same\_0}]^M \tag{12}$$

We calculated Eq. (12) for some cases of P_2, M and τ_2. The computation result is shown in Table 4. The probability that the column values of τ_2 colluders will be the same is very low and decreases as M increases. When $\tau_2 = 3$ and $M = 10$, the probability that colluders are badly chosen is a little high as about 4.3%. However, the probability for

Table 4 Calculation of Eq. (12) for some P_2, M and τ_2

	$M = 10$		$M = 50$		$M = 100$	
	$\tau_2 = 3$	$\tau_2 = 5$	$\tau_2 = 3$	$\tau_2 = 5$	$\tau_2 = 3$	$\tau_2 = 5$
$P_2 = 0.5$	9.54×10^{-07}	9.09×10^{-13}	7.89×10^{-31}	6.22×10^{-61}	6.22×10^{-61}	3.87×10^{-121}
$P_2 = 0.7$	4.81×10^{-05}	2.08×10^{-08}	2.57×10^{-22}	3.86×10^{-39}	6.61×10^{-44}	1.49×10^{-77}
$P_2 = 0.9$	0.042976	0.005155	1.47×10^{-07}	3.64×10^{-12}	2.15×10^{-14}	1.32×10^{-23}

Algorithm 1.

Step 1. ComputeReputation()
Input: $N, M, \{R_j(w_i), i = 1$ to $N,\ j = 1$ to $M\}$
/* Use the method of Bendahmane et al. [20] to compute the reputations of all workers and obtain a set of reputation values */
initialize all R_i, $i = 1$ to N, to zero;
for ($j = 1$; $j \leq M$; $j++$) {
  sort and group $\{R_j(w_:)\}$; // $R_j(w_:)$ is the j-th row of the computation result table
  maj_ans$_j$ = the answer of the majority group for the j-th task;
  for ($i = 1; i \leq N; i++$)
    if($R_j(w_i)$ == maj_ans$_j$) R_i++;
}
return $\{R_1, \ldots, R_N\}$;

Step 2. Grouping()
Input: $\{R_1, \ldots, R_N\}$
/* Use the method of [20] to assign the workers into groups according to their reputation values */
sort and group $\{R_i, i = 1$ to $N\}$ to $\{G_1, \ldots, G_k\}$ so that the reputation of G_k is highest while that of G_1 is lowest;
return $\{G_1, \ldots, G_k\}$;

Step 3. FindHonestGroup()
Input: $\{G_1, \ldots, G_k\}$, $\{R_1, \ldots, R_k\}$, τ_1
if($d = |R_k - R_{k-1}| \geq \tau_1$) return G_k; // highest reputation group
else if($d = |R_2 - R_1| \geq \tau_1$) return G_1; // lowest reputation group
else go to Step 4;

Step 4. FindHonestWorkers()
Input: $\{G_1, \ldots, G_k\}$, $\{R_j(w_i), i = 1$ to $N, j = 1$ to $M\}$, τ_2
$S = \{G_1, \ldots, G_k\}$;
while($S \neq \phi$){
  select G_t from S which $n = |G_t|$ is largest;
  $S = S - \{G_t\}$;
  extract $\{R_:(w_i), i = 1$ to $n\}$ of G_t from $\{R_:(w_i), i = 1$ to $N\}$;
  sort $R_:(w_i), i = 1$ to n; // column vector sorting in Fig. 7.
  /* $R_:(w_i)$ is all computation results of worker i for M tasks. This corresponds to the i^{th} column vector in Fig. 7. */
  for($i = 1; i > n - \tau_2 + 1; i++$){ // Worker ID starts from one. If $i > n - \tau_2 + 1$, remaining workers are less than τ_2.
    $G_{honest} = \{w_i\}$;
    for(count = 1; $i < n$ && $R_:(w_i) == R_:(w_{i+1})$; count++, i++)
      $G_{honest} = G_{honest} \cup \{w_{i+1}\}$; // $R_:(w_i) == R_:(w_{i+1})$ represents column-wise comparison.
    if(count $\geq \tau_2$)
      return G_{honest};
  }
}
return ϕ; // no honest workers are found

other cases is very low. Usually M increases as computation proceeds in cloud servers. Therefore, we set the value of τ_2 to 3.

3.4 Algorithm

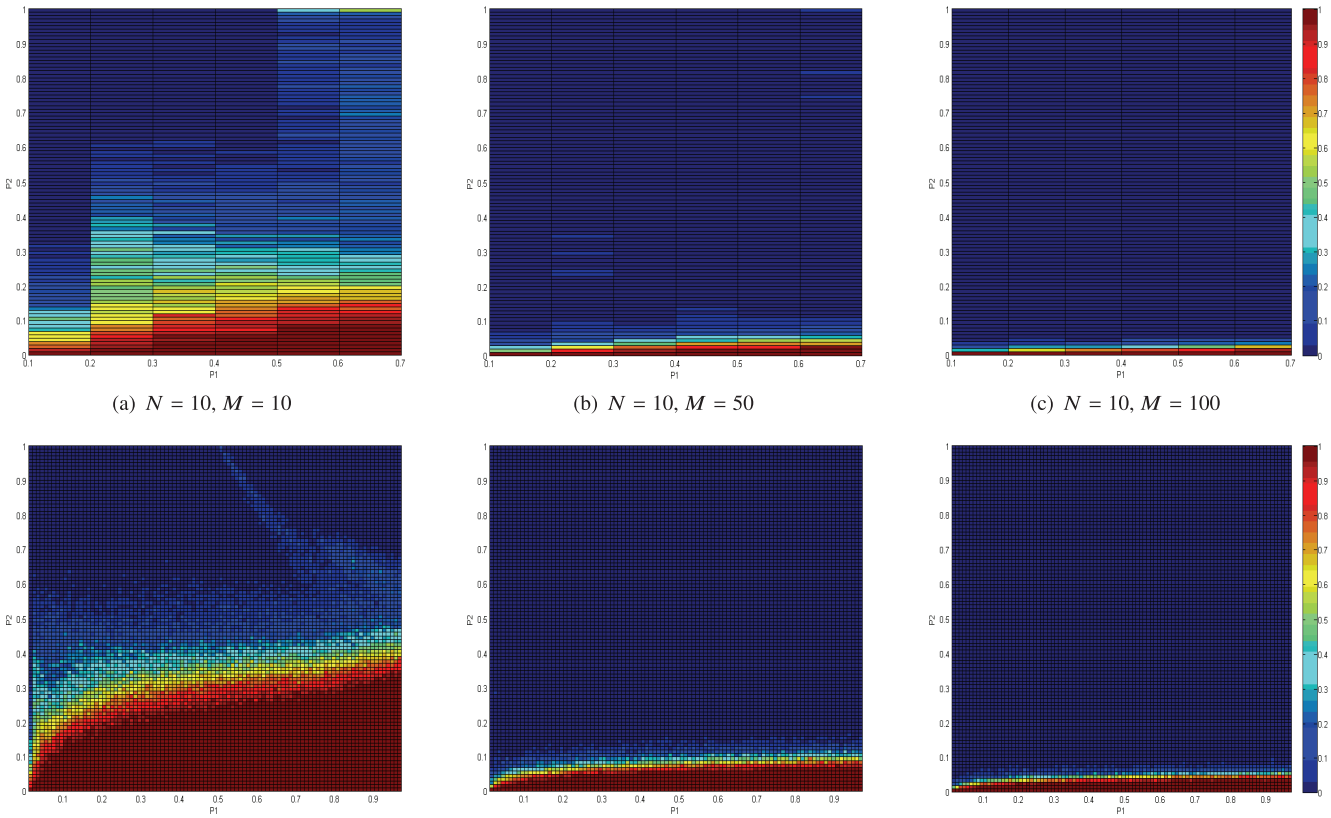
The algorithm used takes the number of workers N, the number of tasks M, and a computation result table containing the computation history $\{R_j(w_i), i = 1$ to $N, j = 1$ to $M\}$ as inputs. To determine honest workers among all workers, the algorithm progresses as shown in Algorithm 1.

Thanks to the theoretical analysis of Sect. 3.2, Algorithm 1 does not require P_1 and P_2 parameters. Therefore, it can be used regardless of the number of colluders.

4. Evaluation

4.1 Accuracy

Some accuracy analyses have been separately conducted for some fixed parameters of N, M, τ_1, and τ_2 in Sect. 3, that is grouping accuracy analysis and identifying accuracy analysis for honest workers in a mixed group. However, in this section, more comprehensive evaluation for Algorithm 1 will be conducted using various parameters of N, M, τ_1, and τ_2. Another definition of a failure used for this evaluation is



(a) $N = 10, M = 10$ (b) $N = 10, M = 50$ (c) $N = 10, M = 100$

(d) $N = 100, M = 10$ (e) $N = 100, M = 50$ (f) $N = 100, M = 100$

Fig. 8 Failure rates with changes in N and M

given below:

Definition 4 (Identifying Failure) *If the honest workers identified are different from the original honest workers that exist in a system, our method is considered failed.*

To improve the reliability of our method, we ran our method 100 times and measured the failure rates. First, we fixed τ_1 at 0.2, and τ_2 at 3. With the changes in N and M, the failure rates were measured as shown in Fig. 8.

Because $\tau_2 = 3$, the range of P_1 was set to 0.1–0.7 for $N = 10$, and 0.01–0.97 for $N = 100$. The range of P_2 was set to 0–1 for all cases. Figure 8 shows that no matter how many workers there are, if the number of tasks is insufficient, the failure rate will be high. This can be explained through Eq. (12) because P_{same} will become higher as M decreases.

As in the case of $M = 10$, if the number of tasks is small, it is shown that the failure rate is more influenced by P_2 (the actual collusion probability of a colluder) than P_1 (the number of colluders divided by N). In the two graphs on the left column of Fig. 8, when P_2 is greater than 0.5 it appears a low failure rate. The failure rate becomes increasingly higher when P_2 is less than 0.5. If malicious workers hide without collusion and primarily return true results, Algorithm 1 misunderstands as if they were honest workers. Therefore, they are not detected well.

P_1 affects the failure rate less than P_2. The higher P_1 is, the region where the failure rate is high slightly increases. This means, if P_2 is constant, the more the colluders are, the more difficult it is for all the colluders to be detected.

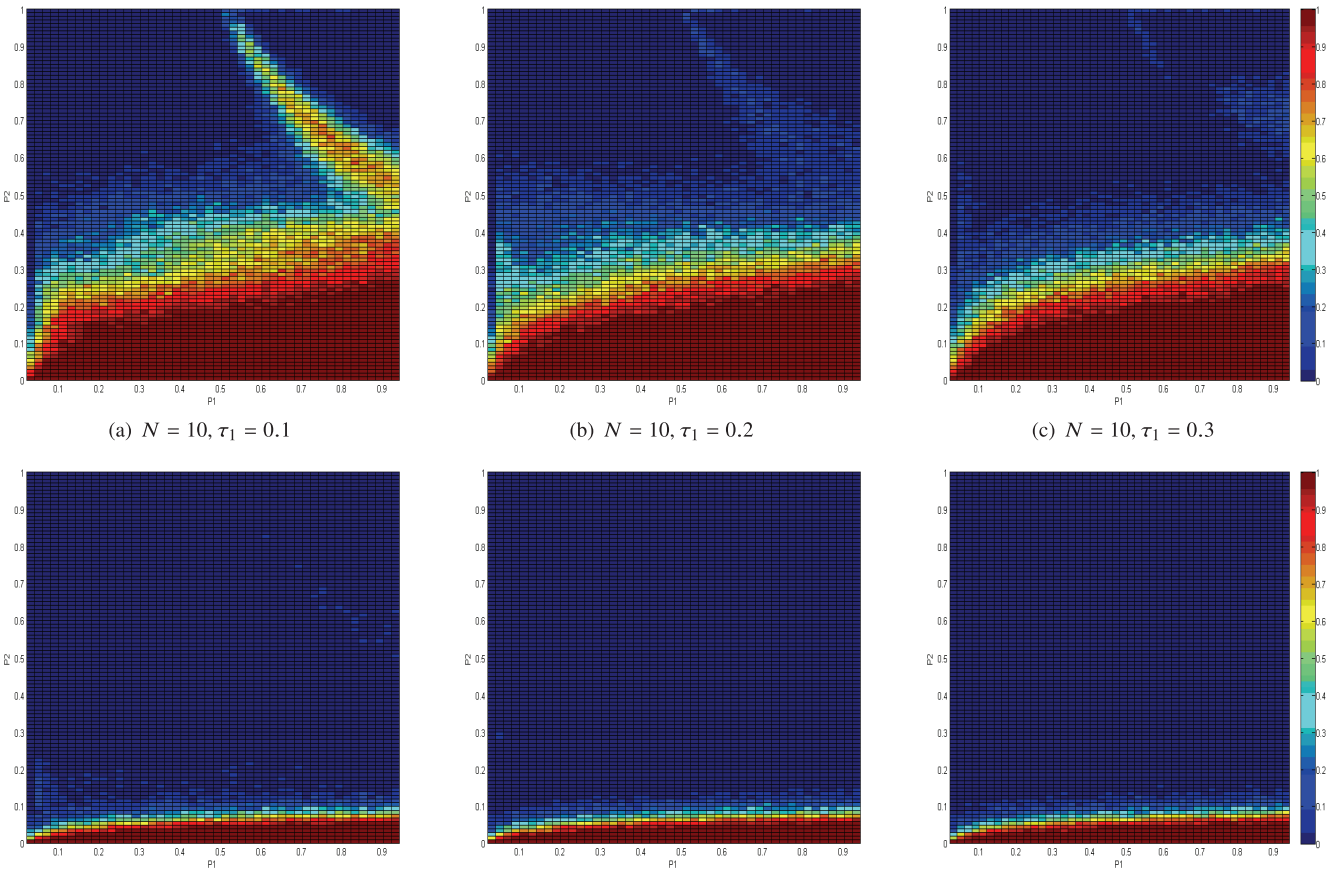
Toward the right column in Fig. 8, it can be seen that the red area is significantly reduced. As the number of tasks (votes) increases, the more elaborated the reputation values are and the better Algorithm 1 finds colluders.

If M is large enough, e.g. 100, our method performs well and can determine all honest workers accurately except when $P_2 < 0.1$. This is because if $P_2 < 0.1$, colluders rarely collude for each vote and mostly act as honest workers.

Next, we fix N at 50, and τ_2 at 3. With the changes in M and τ_1, the failure rates are measured as shown in Fig. 9.

We compared the accuracy by varying τ_1 from 0.1 to 0.3. Figure 9 shows that when the number of tasks is small, the accuracy is slightly influenced by τ_1. In particular, the accuracy is best when $\tau_1 = 0.3$. As in the case of Fig. 8, P_2 gives more influence on the failure rate than P_1. And the higher P_2 is, the higher the failure rate is.

In the case of $M = 10$, $\tau_1 = 0.1$, there is a hornlike red region in the upper right area of the graph. In this region, the failure is high even though P_2 is low. This is the case that the number of colluders is more than half, which is similar to the case of a mixed group analyzed in Sects. 3.1 and 3.2. Grouping errors were found to occur in Step 3 of Algorithm 1 in the case of a smaller τ_1.



(a) $N = 10, \tau_1 = 0.1$ (b) $N = 10, \tau_1 = 0.2$ (c) $N = 10, \tau_1 = 0.3$

(d) $N = 50, \tau_1 = 0.1$ (e) $N = 50, \tau_1 = 0.2$ (f) $N = 50, \tau_1 = 0.3$

Fig. 9 Failure rates with the changes in M and τ_1

If M is smaller, a smaller number of groups are generated and the differences in reputation between adjacent groups will be higher. Therefore, a colluder group is sometimes mistakenly chosen as an honest group. In this case, to judge the significance of the difference, a higher τ_1 will be necessary. $\tau_1 = 0.2$ and $\tau_1 = 0.3$ have no big difference in failure rates.

However, if the number of tasks is large, the accuracy is rarely influenced by τ_1. If the number of task is large, i.e., the higher the number of votes is, the reputation values will be more accurate, so our method will find colluders better regardless of τ_1. Therefore, the determination of τ_1 depends on the value of M. If M is sufficiently large, τ_1 is set as a small value. For example, τ_1 may be set to 0.2. Otherwise, τ_1 is set to a larger value such as 0.3.

4.2 Performance

Step 1 of Algorithm 1 comprises computations for all workers and all voting results. Each majority voting includes sorting and grouping for all workers' computation results, and such votings are performed for M tasks. Thus it is computed in $O(MN \log_2 N)$, where N is the number of workers and M is the number of tasks (votes). Since Step 2 is required to sort all workers and partition them into groups, this step can be computed in $O(N \log_2 N)$. Step 3 needs to compute the difference between the highest and second-highest reputations, or the difference between the lowest and second-lowest reputations, which can be computed in $O(1)$. Step 4 consumes a large amount of time because it has to examine all computation results of the workers. Column-vector sorting is required in a group. If only one group is generated, it can be computed in $O(MN \log_2 N)$.

Usually, more than one group will be generated. In this case, the time complexity of the biggest group is dominant. Therefore, it can be computed in $O(Mn \log_2 N)$, where n is the number of workers in the biggest group. From the above results, it can be concluded that the complexity of Step 4 is much more than that of Step 3. Therefore, using Step 3, we can reduce the execution time of our method.

For further details, we conducted experiments to compare the execution time (10^{-4}s) of Bendahmane et al.'s [20] method with that of our own. Here, we call Bendahmane et al.'s method the existing method for simplicity. The experimental environment is as the same as described in Sect. 4.1.

The red values in Table 5 represent the time cost when our method is executed until Step 4. From Table 5, for a larger M and/or N, the required time also increases for both the existing method and our method, correspondingly. If

Table 5 Execution times of Bendahmane et al.'s [20] method and our own (10^{-4}s).

(a) $N = 10$, $M = 10$, $\tau_1 = 0.3$ and $\tau_2 = 3$

	$P_1 = 0.2$		$P_1 = 0.4$		$P_1 = 0.7$	
	Existing Method	Our Method	Existing Method	Our Method	Existing Method	Our Method
$P_2 = 0.9$	1.1249	1.5828	1.4111	1.6432	1.6550	1.8539
$P_2 = 0.7$	1.0188	1.5926	1.4071	1.7531	1.5230	4.5206
$P_2 = 0.5$	1.0199	1.5829	1.3306	1.7129	1.6092	4.1180
$P_2 = 0.2$	1.1051	2.9189	1.4281	3.0207	1.5199	3.5209

(b) $N = 10$, $M = 100$, $\tau_1 = 0.2$ and $\tau_2 = 3$

	$P_1 = 0.2$		$P_1 = 0.4$		$P_1 = 0.7$	
	Existing Method	Our Method	Existing Method	Our Method	Existing Method	Our Method
$P_2 = 0.9$	4.0288	4.1129	4.3139	4.2132	4.4935	4.8138
$P_2 = 0.7$	3.9514	4.1126	4.2201	4.4131	4.3532	8.1216
$P_2 = 0.5$	4.2852	4.5129	4.3621	4.7129	4.5301	8.6180
$P_2 = 0.2$	4.1235	4.3129	4.3022	4.6130	4.5110	4.9133

(c) $N = 100$, $M = 10$, $\tau_1 = 0.2$ and $\tau_2 = 3$

	$P_1 = 0.2$		$P_1 = 0.4$		$P_1 = 0.7$		$P_1 = 0.9$	
	Existing Method	Our Method	Existing Method	Our Method	Existing Method	Our Method	Existing Method	Our Method
$P_2 = 0.9$	5.0797	5.7135	5.6130	5.7133	5.7933	6.0138	5.8166	6.0139
$P_2 = 0.7$	5.2901	6.1136	5.49179	6.3129	6.2948	12.0188	6.0158	6.2139
$P_2 = 0.5$	5.5190	5.8153	5.6078	6.8159	6.3034	12.0218	6.1986	17.0238
$P_2 = 0.2$	5.2858	12.2208	5.6880	12.4202	5.8766	11.5194	5.7321	11.0196

(d) $N = 100$, $M = 100$, $\tau_1 = 0.2$ and $\tau_2 = 3$

	$P_1 = 0.2$		$P_1 = 0.4$		$P_1 = 0.7$		$P_1 = 0.9$	
	Existing Method	Our Method	Existing Method	Our Method	Existing Method	Our Method	Existing Method	Our Method
$P_2 = 0.9$	11.6490	11.2153	11.9141	12.0150	13.0135	12.8154	13.0137	12.9148
$P_2 = 0.7$	12.1136	12.0147	12.9137	13.3143	14.0137	23.0298	14.8142	15.0156
$P_2 = 0.5$	12.3137	12.2148	13.8145	13.2144	14.3146	14.4147	14.7149	23.5306
$P_2 = 0.2$	11.7137	11.1143	13.3147	13.4147	13.1142	13.0143	13.7140	13.5233

no mixed group exists, such as $P_1 = 0.2$ and $P_2 = 0.9$, or $P_1 = 0.4$ and $P_2 = 0.7$, the time cost of our method is almost the same as that of the existing method. If a mixed group does exist, such as $P_1 = 0.7$ and $P_2 = 0.7$, or $P_1 = 0.9$ and $P_2 = 0.5$, the time cost increases considerably because our method has to run to Step 4. As shown in Figs. 1 and 2, in many cases, no mixed groups are generated. Therefore, the time cost and complexity of our method are almost the same as those of the existing method for many cases.

5. Conclusion

The traditional reputation-based method was proposed based on the assumption that the number of colluders is less than half of all workers. Our study enabled to weaken this assumption by investigating the accuracy of the grouping method. The study discovered that the grouping accuracy is indeterminate and changes with the number of colluders and their collusion probability. The most valuable result of the study is: If a group has a significantly different reputation value from other groups, the group is the honest group. Even a group with the lowest reputation value can be the honest group if the number of colluders is much more than that of honest workers.

Based on this study, we proposed a new method to identify honest workers from a mix of colluders and honest workers. When no mixed groups are generated, we can determine the honest group quickly by computing the difference in reputation between groups and set threshold, τ_1. Otherwise, to determine honest workers, we check the voting results of all groups one by one and set another threshold, τ_2. Thanks to a rigorous mathematical analysis, our algorithm can detect colluders without prior knowledge about the number of colluders and their collusion probability.

Our experiment and analysis results show that our proposed method can be used to efficiently identify honest workers and guarantees high accuracy even without restrictions under the assumption that the number of colluders is less than half of all workers. To improve the accuracy, we may attempt to increase the number of tasks and vary the values of thresholds τ_1 and τ_2. Our method also performs well when these values vary sufficiently, even under extreme conditions in which few or a large number of colluders exist. As a way to find honest workers, it would be interesting to integrate our proposed method into an existing cloud computing system.

Acknowledgments

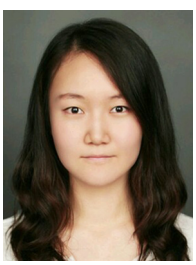
This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant (No. R0190-15-2011, Development of Vulnerability Dis-

covery Technologies for IoT Software Security), and the National Research Foundation of Korea grant (NRF-2014R1A2A2A01005519 and NRF-2014R1A1A2056266) funded by the Korea government (MSIP).

References

[1] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and privacy in cloud computing: A survey," in Sixth International Conference on Semantics Knowledge and Grid (SKG), pp.105–112, Nov. 2010.
[2] Z. Xiao and Y. Xiao, "Security and Privacy in Cloud Computing," IEEE Communications Surveys & Tutorials, vol.15, no.2, pp.843–859, July 2013.
[3] H.W. Wang, "Integrity verification of cloud-hosted data analytics computations," in Proceedings of the 1st International Workshop on Cloud Intelligence, Article no.5, 2012.
[4] Y. Wang and J. Wei, "VIAF: Verification-Based Integrity Assurance Framework for MapReduce," in IEEE International Conference on Cloud Computing (CLOUD), pp.300–307, July 2011.
[5] L. Zhang, Q. Li, Y. Shi, L. Li, and W. He, "An integrity verification scheme for multiple replicas in clouds," Web Information Systems and Mining, Lecture Notes in Computer Science, vol.7529, pp.264–274, 2012.
[6] L.F.G. Sarmenta, "Sabotage-tolerance mechanisms for volunteer computing systems," Future Generation Computer Systems, vol.18, no.4, pp.561–572, 2002.
[7] S. Zhao, V. Lo, and C.G. Dickey, "Result verification and trust-based scheduling in peer-to-peer grids," in Fifth IEEE International Conference on Peer-to-Peer Computing, pp.31–38, Sept. 2005.
[8] A.C. Oliveira, L. Sampaio, S.F. Fernandes, and F. Brasileiro, "Adaptive Sabotage-Tolerant Scheduling for Peer-to-Peer Grids," in Fourth Latin-American Symposium on Dependable Computing, pp.25–32, Sept. 2009.
[9] P. Domingues, B. Sousa, and L.M. Silva, "Sabotage-tolerance and trust management in desktop grid computing," Future Generation Computer Systems, vol.23, no.7, pp.904–912, Aug. 2007.
[10] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao, "Cluster computing on the fly: P2P scheduling of idle cycles in the internet," in Peer-to-Peer Systems III, vol.3279, pp.227–236, 2004.
[11] K. Watanabe, M. Fukushi, and S. Horiguchi, "Collusion-resistant sabotage-tolerance mechanisms for volunteer computing systems," in IEEE International Conference on e-Business Engineering, pp.213–218, Oct. 2009.
[12] Y. Ding, H. Wang, L. Wei, S. Chen, H. Fu, and X. Xu, "VASW: Constructing trusted open computing system of MapReduce with verified participants," IEICE Transactions on Information & Systems, vol.E97-D, no.4, pp.721–732, 2014.
[13] J. Sonnek, A. Chandra, and J. Weissman, "Adaptive reputation-based scheduling on unreliable distributed infrastructures," IEEE Transactions on Parallel and Distributed Systems, vol.18, no.11, pp.1551–1564, Nov. 2007.
[14] E. Staab and T. Engel, "Collusion detection for grid computing," in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.412–419, 2009.
[15] M. Moca, G.C. Silaghi, and G. Fedak, "Distributed Results Checking for MapReduce in Volunteer Computing," in IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), pp.1847–1854, May 2011.
[16] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation Systems," Communications of the ACM, vol.43, no.12, pp.45–48, Dec. 2000.
[17] B. Alunkal, I. Veljkovic, G. von Laszewski, and K. Amin, "Reputation-Based Grid Resource Selection," Workshop of Adaptive Grid Middleware (AGridM), Sept. 2003.
[18] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks," in Proceedings of the Twelfth International World Wide Web Conference, pp.640–651, 2003.
[19] H.Y. Lee, J.W. Kim, and K. Shin, "Simplified Clique Detection for Collusion-resistant Reputation Management Scheme in P2P Networks," in International Symposium on Communications and Information Technologies (ISCIT), pp.273–278, Oct. 2010.
[20] A. Bendahmane, M. Essaaidi, A.E. Moussaoui, and A. Younes, "Result verification mechanism for MapReduce computation integrity in cloud computing," in International Conference on Complex Systems (ICCS), pp.1–6, Nov. 2012.
[21] H.-C. Tsai, N.-W. Lo, and T.-C. Wu, "A threshold-adaptive reputation system on mobile ad hoc networks," IEICE Transactions on Information & Systems, vol.E92-D, no.5, pp.777–786, 2009.
[22] E. Abdullah and S. Fujita, "Reputation-based colluder detection schemes for peer-to-peer content delivery networks," IEICE Transactions on Information & Systems, vol.E96-D, no.12, pp.2696–2703, 2013.
[23] J. Sonnek, M. Nathan, A. Chandra, and J. Weissman, "Reputation-based scheduling on unreliable distributed infrastructures," in 26th IEEE International Conference on Distributed Computing Systems, p.30, 2006.

Junbeom Hur received the B.S. degree in computer science from Korea University in 2001, the M.S. and Ph.D. degrees in computer science from KAIST in 2005 and 2009, respectively. He has been in the University of Illinois at Urbana-Champaign as a postdoctoral researcher from 2009 to 2011, and in the Chung-Ang University in Korea as an assistant professor from 2011 to 2015. He is currently an assistant professor in the department of computer science and engineering at the Korea University in Korea. His research interests include information security, mobile computing security, cyber security, and applied cryptography.



Mengxue Guo received the BS degree from Huaiyin Normal University, China, and Konkuk University, Seoul, Korea in 2012 and the MS degree in Electronics and Electrical Engineering from Chung-Ang University, Seoul, Korea in 2014. Her research interests include cryptology and cloud computing security.

Younsoo Park received the BS degree from Chung-Ang University, Seoul, Korea in 2013. He is currently pursuing his MS degree in School of Electrical and Electronics Engineering, Chung-Ang University, Seoul, Korea. His research interests include big data processing, data analytics and internet security.

Chan-Gun Lee received the BS, MS, and PhD degrees in computer science from Chung-Ang University, KAIST, and the University of Texas, Austin, in 1996, 1998, and 2005, respectively. From 2005 to 2007, he was a senior software engineer at Intel. Currently, he is an associate professor of computer science and engineering at Chung-Ang University. His research interests span real-time systems, software engineering for time-critical systems, test frameworks.

Ho-Hyun Park received the BS degree from Seoul National University, Seoul, Korea in 1987 and the MS and Ph.D. degrees from KAIST in 1995 and 2001, respectively. From 1987 to 2002, he worked at Samsung Electronics as a principal engineer. He is currently a professor in School of Electronics and Electrical Engineering, Chung-Ang University, Seoul, Korea. His research interests include multimedia processing, big data and internet security.