Computing and Informatics, Vol. 37, 2018, 1293–1312, doi: 10.4149/cai_2018_6_1293

NEW METHODS OF UNCHEATABLE GRID COMPUTING

Jianhua Yu

School of Mathematical Sciences, South China Normal University
Guangzhou, P. R. China
e-mail: yujianhuascnu@126.com

Yuan Li

Department of Mathematics, Winston-Salem State University
NC 27110, USA
e-mail: liyu@wssu.edu

Abstract. Grid computing is the collection of computer resources from multiple locations to reach a common goal. According to the task publisher's computing power, we will classify the deceptive detection schemes into two categories, and then analyze the security of deceptive detection schemes based on the characteristics of computational task function. On the basis of double check, we proposed an improved scheme at the cost of time sacrifice called the secondary allocation scheme of double check. In our scheme, the security of double check has been greatly strengthened. Finally, we analyzed the common problem of High-Value Rare Events, improved the deceptive detection scheme due to [1], and then put forward a new deceptive detection scheme with better security and efficiency. This paper is revised and expanded version of a paper entitled "Deceptive Detection and Security Reinforcement in Grid Computing" [2] presented at 2013 5$^{\text{th}}$ International Conference on Intelligent Networking and Collaborative Systems, Xi'an city, Shanxi province, China, September 9–11, 2013.

Keywords: Distributed, deceptive detection, ringers, grid computing

Mathematics Subject Classification 2010: 68Q85

1 INTRODUCTION

The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. Grids are a form of distributed computing whereby a "super virtual computer" is composed of many networked loosely coupled computers acting together to perform large tasks. Grid computing combines computers from multiple administrative domains to reach a common goal to solve a single task, and may then disappear just as quickly. By grid computing, we can use the computer's idle capacity of tens of thousands of volunteers from all over the world through the Internet and analyze the electrical signal from outer space and find the hidden black holes, at the same time, explore the possible existence of alien wisdom life; still, we can look for more than 10 million digital mason prime numbers; and we can also search for and find more effective drugs to against HIV and to complete the project which needs surprisingly large amount of calculation.

Past few years have seen a tremendous growth in grid computing with its effect being felt in the biotechnology industry, entertainment industry and financial industry, etc. [3, 4, 5, 6, 7, 8, 9]. For example, the Search for Extra-Terrestrial Intelligence (SETI@home) project [6], which distributes to thousands of users the task of analyzing radio transmissions from space, has a collective performance of tens of teraflops. Another Internet computation, the GIMPS project directed by Entropia.com, has discovered world-record prime numbers. Future projects include global climate modeling and fluid dynamics simulation. There are also cryptographic protocols which allow to implement provably optimal systems in theory [10, 11, 12, 13, 14]. However, these algorithms are often computationally expensive in practice.

With the number of participants who join the grid computing increases, inevitably, there will be a lot of cheating. To gain more profits, some people may provide some false certificates to prove that he/she himself/herself has provided a lot of resources, including the computing time or the computing power, etc.; and there are also some people who will directly return some false results. In addition, some people can get more useful information from the calculated target. For example, in the search for an effective approach of grid computing, a participant finds an effective treatment in the grid computing. However, he does not provide it to the supervisor of the grid computing, but sell it to another hospital, in order to gain more profits. Therefore, we need to deal with two things in the grid computing:

1. How to detect the participant cheating;
2. In certain cases, how to hide the results which the task publisher really needs.

2 RELATED WORK

With the rapid development of grid computing, the cheating behavior in the grid computing received more and more attention, and there exists many research achievements [1, 11, 15, 16, 17, 18]. In order to detect the behavior of fraud in the

grid computing, Golle and Mironov proposed ringers scheme [15], which can protect against coalitions of lazy cheaters provided that the computational tasks all involve the Inversion of a One-Way Function (IOWF) f for a given value y, as in the distributed.net attacks on cryptographic functions. In the ringer scheme, during the initialization stage for each participant, the supervisor randomly selects several inputs x_i that will be assigned to that participant and computes $f(x_i)$ for each one. Then, in addition to the value y that the supervisor wishes to invert, the supervisor also sends to that participant all the "ringers" the supervisor has computed for him. The participant must report the pre-images of all the ringers (as well as the pre-image of y if he was lucky enough to discover it). That is, the participant needs to compute f on x for all x in his input domain D and return the pre-image of y if found, and he also has to return all the ringer pre-images he finds. By remembering the ringers for each participant, the supervisor can easily verify whether each participant has found all his ringers or not. If he has, then the supervisor is assured with reasonable probability that the participant has indeed conducted all his computations. In the ringers scheme, the supervisor mixes some previous computation results with the corresponding input and sends them to the participant. Golle and Mironov have confirmed that in Ringers scheme, the participant's cheating success opportunity is very small, but the use of computational task function must be one-way. If participants collaborate with each other, they will know the number of the ringers in the scheme. Therefore, Golle and Mironov proposed the subsequent Hybrid Scheme.

Szajda et al. extend the ringers scheme [17] to deal with other general classes of computations, including optimization and Monte Carlo simulations. They propose effective ways to choose ringers for those computations. To find wrong results created by lazy participants or cheaters, today many systems simply replicate the work units and use the majority rules to decide the correctness of results [20]. However, this approach often implies a waste of CPU cycles. For tasks where the verification is less expensive than redoing the computations, alternative approaches are preferable.

Golle and Mironov [15] propose a solution where a server secretly precomputed the results for a set of input values, the so called ringers. These ringers are then interspersed among the ordinary input values of a work-unit. If a lazy client does not compute the entire unit, it is likely to miss a ringer which can easily be detected by the server. The disadvantages of this approach are the need for precomputation, the redundant computation that inherently occurs, and the fact that a cheater is still likely to be undetected when cheating on a very small fraction of input values only. Szajda et al. have generalized this approach in [17].

Du et al. [18] present a commitment-based sampling scheme for cheater detection in grid computations based on Merkle trees. A server selects some samples which are assigned to a participant. The participant has to commit to its results which are subsequently checked. The drawback of their approach is the additional burden on the server which has to recompute some work-units itself.

In [1] Du and Goodrich introduce the deceptive detection scheme named Searching for high-value rare events. In their criterion-expansion scheme, under the circumstance of one-to-one correspondence in the computational task function, it has very good safety, and it can defeat three kinds of cheating models put forward in this article, but under the circumstance of more-to-one correspondence in the computational task function, the participant will be able to position the supervisor to find the value of the data according to the characteristics of some of the data. In this case, the solution cannot resist the second and the third deceptive model in this article. At the same time, the criterion-expansion detection scheme cannot resist the collusion of the participants. In the criterion-reduction scheme of [1], it lacks the necessary honesty testing steps, and it even cannot judge whether the participant honestly computes all the data on the mission. Hence, there will be some serious defects in terms of security.

Based on the scheme in [1], we present a new scheme using the Monte Carlo simulation method. This scheme can resist the semi-honest cheater and hoarding cheater. This scheme is available for one-to-one function and more-to-one function. At the same time, it can resist the semi-honest cheater and hoarding cheater. But most of schemes are available for one-to-one function.

We also design an other new scheme. In the new scheme, we add some variables which satisfy the criterion Y to the domain of the function in the computational task. For example, we are looking for signs of life in outer space in grid computing. Since the value of the sign of life in outer space data is very very rare, the participant can easily judge the criterion Y according to these characteristics, and therefore cannot hide it. If we add appropriately some biological signs of life on earth and make the domain of Y expand appropriately, thus let a rare event become not rare, and even if participants find the valuable data to satisfy the standard of value Y, they are still unable to distinguish whether these data come from outer space or from earth. Thus, it protects rare events.

In most of cheating detection schemes, the computation task X had not been dealt with. In order to achieve the purpose of hiding Y, they only used new criterion Y' to replace Y. We present a new method. Let Y be y, we can construct a new set C, for $\forall x \in C, f(x) = y$. In this way, we expand the task to $X \cup C$. Compared with[1], this new scheme has greatly improved the security.

3 THE MODEL

3.1 Problem Definition

In grid computing, when a supervisor releases a computational task, he/she needs to compute $f(x)$, $\forall x \in X = \{x_1, x_2, \ldots, x_n\}$ and finally picks out valuable data from the results of x_i and $f(x_i)$. Due to the limited computation ability of the supervisor, he/she will assign tasks to n participants. The task set of every P_i $(i = 1, 2, \ldots, m)$ is D_i. Finally, each participant will return the valuable data to the supervisor in their respective task set.

Here we will introduce some basic terminology in the grid computing.

- Honesty Ratio r: For $\forall x \in D$, the participant needs to compute $f(x)$. But in fact, the participant may only compute $f(x)$ for $x \in D' \subset D$. Define honesty ratio r as $\frac{|D'|}{|D|}$.
- Honesty Participant Ratio p: In grid computing, the ratio of the honest participants in all of the participants should be considered.
- Valuable Events: Data that the supervisor is interested in.
- Honesty Return Ratio h: h is the proportion of correct valuable events which were returned by P_i.
- Criterion y: The criterion can be a specific value or a range, and it can also be a literal description language, etc. Using it, the supervisor and participants will check whether $f(x)$ satisfies the set of the valuable events.
- Pseudo-Valuable Events: The supervisor selects some data which satisfy the criterion to hide the true criterion.
- High-Value Rare Events: The valuable data that satisfy the supervisor's criterion Y is rare, and it even does not exist. For example, in the process of searching for rare blood type, the rare blood type is the rare event.
- Ringer: A ringer is a value chosen by the supervisor in the domain of f.
- $Pr(r)$: Assume that the participant is assigned a task that consists of computing $f(x)$ for all $x \in D$, where $D = \{x_1, \ldots, x_n\}$. If a participant computes the function f only on $x \in D'$, where $D' \subseteq D$, we define the honesty ratio r as the value of $\frac{|D'|}{D}$. When the participant is fully honest, the honesty ratio is $r = 1$; otherwise $r < 1$.

Let $Pr(r)$ be the probability that a participant with honesty ratio r can cheat without being detected by the supervisor. Let $C_{cheating}$ be the expected cost of the required task. We say a grid computing is uncheatable if one of the following or both inequalities are true:

$$Pr(r) < \varepsilon, \text{ for a given } \varepsilon(0 < \varepsilon \leq 1) \text{ or } C_{cheating} > C_{task}.$$

In grid computing, what the supervisor mainly faces is how to ensure the following four conditions established:

1. Every participant P_i will compute $f(x)$ for all data in D_i;
2. Every participant P_i cannot know which data is the valuable data that the supervisor is looking for;
3. The supervisor is able to distinguish the value of the returned data correctly;
4. Even if the participants act in collusion with each other, they cannot break the effective implementation of the task.

In grid computing, the organizer of the computation is the supervisor, at the same time there are a lot of untrusted participants. The Supervisor will allocate computing tasks.

The supervisor can distribute one or more computing tasks $X = \{x_1, x_2, \ldots, x_n\}$ according to the actual situation. The computing task of P_i is D_i, and all of them is $X = D_1 \cup D_2 \cup \ldots \cup D_n$.

For a specific computation, it is performed by two functions described below.

- A computational task function $f : X \to T$ defined on a finite domain X. The goal of the computation is to evaluate f on all $x \in X$. For the purpose of distributing the computation, the supervisor partitions X into subsets. The evaluation of f on subset D_i is assigned to participant P_i.
- A screening function S. The screener is a function that takes as input a pair of the form $((x, f(x)); y)$ for $x \in X$, and returns a string $s = S((x, f(x)); y)$, where y represents the criterion. S is intended to screen for "valuable" outputs of f that are reported to the supervisor by means of the string s.
- A payment scheme P. The payment scheme is a publicly known function P that takes as input a string s from participant i and outputs the amount due to that participant. We require that P may be efficiently evaluated. Specifically, one evaluation of P should equal a small constant number of evaluations of f. In many articles, they use the P function, but we do not use it.

3.2 Three Deceptive Models

This article uses three similar kinds of deceptive models proposed by Du and Goodrich [1]. We assume each participant is given a domain $D \subset X$, and his/her task is to compute $f(x)$ for all $x \in D$.

1. **Semi-Honest Cheater Model.** In this model, the participant follows the supervisor's computations with one exception: for $x \in \breve{D} \subset D$, the participant uses $\breve{f}(x)$ as the result of $f(x)$. Function $\breve{f}$ is usually much less expensive than function f; for instance, $\breve{f}$ can be a random guess. The goal of the cheating participant in this model is to reduce the amount of computations, such that it can maximize its gain by "performing" more tasks during the same period of time.
2. **Hoarding Cheater Model.** In this model, the participant conducts all the required computations. However, the participant will keep the computation results if the results are valuable. However, the participant will keep the computation results if the results are valuable. For example, if the computation is to search for a rare event, a "lucky" participant who has found a rare event might report a negative result because of the value of such a result. This type of cheating behavior is a cheating on the screening function S.
3. **Malicious Cheater Model.** In this model, the behaviors of the participant can be arbitrary, or even be hostile. For example, the participant P does all the

correct calculation $f(x)$ to all of the data in D, but he deliberately returns to the supervisor wrong value screening results, to achieve the purpose of disorder and confuse the supervisor to work normally, the participant may be the supervisor of the competitors.

4 DECEPTIVE DETECTION MODEL

Whether the supervisor has the computing power is decided the way they test cheating. On this basis, we divide the deceptive detection model into two forms.

4.1 The Supervisor Has Not Computing Power

Because the supervisor has not ability to compute, so it is given a task data x, the supervisor is unable to determine the accuracy of $f(x)$. The supervisor can only verify according to the participant. For this case, we can adopt the method of double check.

Kuhn et al. [21] consider a grid framework consisting of a server and a potentially large number of clients. A client is the logic entity with which the server interacts. The server sends its work-units (or tasks or jobs) to clients, which return the corresponding set of results. A participant is a user who has registered an account for the project. He/she may use one or more computers (or machines) working for him/her in the project. Moreover, a computer can correspond to one or several clients. The computational resources of the clients are heterogeneous.

The server distributes two different kinds of tasks to the clients. Work-units are the main computational tasks of the grid computing framework; checking units require the client to perform a number of checks for the different results. They assume that the main incentive for participation in the project are credit points: There may be websites listing the credit points earned by the different users, or there may even be ways to convert credit points into real money, by a lottery, for example. In their system, a client earns credit points for computing both work-units and checking units. The number of points is thereby proportional to the amount of work, such that a participant is indifferent between the two tasks.

It is of prime importance that the credit points be earned honestly. Their algorithms' goal is to make sure that participants only get the credits they really deserve, and that the system is not flooded with wrong results. They distinguish between two kinds of clients: good clients and bad clients (or cheaters). They consider a harsh model where all cheaters form a single coalition. Today's systems such as Seti@Home are reported to have roughly 1 % cheaters.

A checking algorithm which identifies dishonest behavior can achieve a higher effectiveness if it interplays with a mechanism to punish cheaters. They assume that a wrong result or an improper check implies that the corresponding client is a cheater. Being debunked as a cheater basically implies that the corresponding client's user loses all its credits, and the corresponding account is closed.

1. **Double-Check.** The supervisor will assign the subtask to two or more different participants, and then compare whether the returned results are consistent with each participant's. If it is consistent, the supervisor believes that the participant is honest, otherwise, allocates the task again.

   Security Analysis. If there is no collusion between participants, double check scheme has a good security; if the participants of the same task are in collusion with each other, they will return the same results to the supervisor. The supervisor will still believe they are honest participants. So double check scheme cannot defeat the joint attack of the participants.

   In order to resist the joint attack, we can adopt the deformation mode of double check.

2. **Deformation 1.: The secondary allocation in double check scheme.** The supervisor will randomly assign the subtask D to the participant A firstly. After it returns the results, the supervisor will randomly assign the same task D to another participant B. For determining honesty of the participant A and B, the supervisor compares conformance of the returned data from A and B.

3. **Deformation 2: The samples testing in double check scheme.** The computing task X is divided into n parts, i.e. $X = D_1 \cap D_2 \ldots \cap D_n$. The participant P_i has computing task D_i. The supervisor randomly selects $d_i \in D_i$ as the honesty sample data of P_i. When all the results are returned, the supervisor will assign $d_1, d_2, \ldots, d_n$ randomly to n new participants (at this stage, adopting the secondary allocation in double check scheme to verify these n new participants' honest behaviors). The supervisor will compare the sample data with returned values of P_1, P_2, ..., P_n, if the returned value of P_j is consistent with the sample data, the participant is honest.

   Security and efficiency analysis.

1. The two deformations of double check improve the security of the scheme. Because A does not know another participant B, he will compute honestly. Otherwise, the supervisor will know he is a dishonest participant.

2. Because the samples of double check are only a small part of X, the deformation 2 which is relative to the former has a very good improvement in efficiency.

3. Both double check and two deformations require higher honest proportion p, otherwise progress of completing the task is too slow, the price is too big. Because the supervisor needs to repeat each computing task of data distribution, computing resources will be wasted.

4. The two deformation schemes in security have improved, but at the same time caused the stagnation on time.

4.2 The Supervisor Has Computing Power

The supervisor has a certain ability to compute $f(x)$, but this kind of ability is limited. Because of the large workload, the supervisor cannot independently complete all the verification.

[18] puts forward a kind of uncheatable grid computing based on binary tree. When the supervisor uses m samples, the participant which honesty ratio is r can deceive the chance of success

$$Pr(cheating - succeeds) = [r + (1 - r)q]^m,$$

for $x \in D - D^{'}$, q is the probability that the participant correctly guesses the value of $f(x)$.

Combining both cases of $x \in D^{'}$ and $x \in D - D^{'}$, for one sample x, the probability that the participant can prove its honesty on sample x is $(r + (1 - r)q)$. Therefore, the probability that the participant can prove its honesty on all m samples is $(r + (1 - r)q)^m$.

To keep the probability of successful cheating below a small threshold ε, the sample size m should be

$$m \geq \frac{\log \varepsilon}{\log(r + (1 - r)q)}.$$

But this detection method can only detect whether the participant computes correctly the $f(x)$, $\forall x \in D$. The participant will still be able to deceive the screener function. So anti-cheat binary tree method can only detect semi-honest cheater.

5 THE HIDDEN CRITERION

The supervisor wants to search for valuable data from a large database. He can adopt the method of grid computing and ask participants to help to search. The supervisor has the criterion Y, but he cannot disclose it to participants. Otherwise the participant will obtain valuable data. From thousands of drugs, for example, the supervisor wants to find a formulation for the treatment of certain disease effectively, but does not want to let participant know effective standard of the formulation.

In order to avoid leaking the criterion Y, the supervisor constructs a new criterion to replace Y. So the supervisor can hide Y and the valuable data $x \in \{x \mid f(x) = Y, x \in D\}$. In general, there are two kinds of typical processing method for criterion Y.

5.1 Criterion Expansion

Golle and Mironov put forward the basic ringer scheme [15]. The basic ringer scheme hides the criterion by the method of criterion-expansion method. Let the supervisor's criterion be $Y = \{y\}$, we are looking for valuable x which satisfies $f(x) = y$, $x \in D$. The supervisor and participant will perform the following steps.

1. The supervisor chooses for participant P_i uniformly independently at random n values x_1^i, x_2^i, ..., x_n^i in D_i, and also computes the corresponding images: $y_j^i = f(x_j^i)$.
2. The screener S_i is defined as follows. On input $(k, f(k))$, test whether $f(k)$ belongs to the set $\{y, y_1^i, y_2^i, \ldots, y_n^i\}$. If so output the string k, otherwise output the empty string.
3. The secret key K_i is the set x_1^i, x_2^i, ..., x_n^i, which we call the set of ringers.
4. The supervisor checks that s_i contains all the ringers in K_i plus possibly x such that $f(x) = y$. If so, the participant is honest.

Security Analysis.

1. If the criterion y included in the set $\{y, y_1^i, y_2^i, \ldots, y_n^i\}$ is not hidden, participant P_i can relatively easy to determine y.
2. Only for f is a one-way function. If f is not a one-way function, the participant can solve the preimages set of x_1^i, x_2^i, ..., x_n^i directly from the set of $\{y, y_1^i, y_2^i, \ldots, y_n^i\}$ and does not need to compute tasks in each function value $f(x)$. But the supervisor cannot find any cheating behavior.
3. Participant P_i knows the number of the elements in the ringer set. If participant P_i found all elements in the ring set, he needs not to do the rest of the calculation.
4. This scheme cannot prevent collusion. By comparing $\{y, y_1^i, y_2^i, \ldots, y_n^i\}$ with $\{y, y_1^j, y_2^j, \ldots, y_n^j\}$, P_i and P_j can gain the criterion y and its corresponding valuable x.

After the basic ringer scheme, Golle and Mironov put forward two improvement schemes: Bogus ringer scheme and Hybrid scheme.

5.2 Criterion Reduction

Assuming the supervisor has computing task $f(x), x \in X$ and criterion $Y = \{y \mid y = f(x)$, and satisfies the constraints of $y_1, y_2, \ldots, y_t\}$.

1. The supervisor partitions all participants into n parts ($n < t$, n is safety parameter). Each part has the ability to complete computing $f(x)$, $x \in X$ alone.
2. The supervisor distributes computing task $f(x)$, $x \in X$ and criterion $\{y_i\}$ to part i. Part i eventually returns the set S_i, for $\forall x \in S_i$, $y_i = f(x)$.
3. The supervisor gets the set $S = S_1 \cap S_2 \cap \ldots \cap S_n$. He can verify each participant's behavior.
4. The supervisor's criterion is $y' = \{y \mid f(x) = y, x \in S$ and satisfies the constraints of $y_{n+1}, y_{n+1}, \ldots, y_t\}$.

In the process of allocating the task by supervisor, just sent t constrained conditions to n parts, even if all participants are in collusion with each other, they also cannot get to judge criterion Y.

Because of $|S| \ll |X|$, the supervisor can easily gain the valuable data.

6 SEARCHING FOR HIGH-VALUE RARE EVENTS

Du and Goodrich [1] introduce the searching for high-value rare events scheme.

They propose a more practical set of grid computations – *data filtering* problems. In data filtering problems they are given a large set X of data instances and a Boolean filtering function f. The supervisor is interested in all the elements x of X such that $f(x) = 1$. Usually, the function f will involve some internal scoring function on each input x along with a threshold value such that if x scores above this value, then x is considered rare and interesting. This class of problems includes the SETI@home application, where X consists of extraterrestrial signals that are scored against what are considered to be patterns of intelligence.

By their very nature, it is not obvious which of the inputs in X will score positive for the filter f (for otherwise there would be no motivation for them to go to the trouble of using a grid computing environment to solve this problem). For example, a casual examination of the signals that have scored highest so far in the SETI@home scoring function does not yield any obvious patterns; to the naked eye they all appear as noise. Thus, for data filtering applications such as this, employing an input chaff injection scheme is easy.

To inject input chaff into the set of tasks, the supervisor needs only to have a set of instances Y such that determining if any member y_i is not in X is at least as difficult as computing $f(y_i)$. (The supervisor may not need to explicitly construct Y if he/she has a way of choosing elements from Y probabilistically.) Then the supervisor can randomly inject members of Y into the task sets $D \subset X$ for each participant (with some probability p) to provably obfuscate the rare events. For example, a true input x in the SETI@home application could be transformed into chaff simply by adding a pattern of intelligence to it.

1. The supervisor randomly selects m inputs $x_1, \ldots, x_m$ from the input domain X. Note that X is the global input domain, each participant only conducts tasks for a subset of X.
2. The supervisor generates m chaff by computing $c_i = \text{hash}(f(x_i))$, for $i = 1, \ldots, m$.
3. The supervisor sends the list $C = \{\text{hash}(y), c_1, \ldots, c_m\}$ to all the participants. C should be permuted to hide hash(y).
4. For any input x assigned to each participant, the participant computes hash$(f(x))$ and compares the results with the list C. If a match occurs, the participant sends x back to the supervisor; otherwise x is discarded.

5. The supervisor can verify whether a returned x value is an actual rare event or chaff, by a simple lookup in C (say, by storing the elements of C in a hash table). The supervisor also checks whether the participant whose tasks include chaff has returned the chaff or not. This way, the cheater can be caught.

This scheme cannot resist collusion of participants in a criterion expansion type and cannot detect the semi-honest cheater in a detection scheme. Here we introduce two schemes in [1].

6.1 Scheme I (Criterion Expansion)

1. The supervisor randomly selects n inputs $x_1, \ldots, x_n$ from the input domain X. Note that X is the global input domain, each participant only conducts tasks for a subset X.
2. The supervisor generates n ringers by computing

$$c_i = \text{hash}(f(x_i)),$$

for $i = 1, \ldots, n$.

3. The supervisor sends the list

$$C = \{\text{hash}(y), c_1, \ldots, c_n\}$$

to all the participants. C should be permuted to hide hash(y).

4. For any input x assigned to each participant, the participant computes hash$(f(x))$ and compares the results with the list C. If a match occurs, the participant sends x back to the supervisor, otherwise x is discarded.
5. The supervisor can verify whether a returned x value is an actual rare event or ringer, by a simple lookup in C (say, by storing the elements of C in a hash table). The supervisor also checks whether the participant whose tasks include ringer has returned the ringer or not. This way, the cheater can be caught.

If f is a one-to-one function, this scheme has a good security. Even if all participants in the scheme are in collusion with each other, we can also hide criterion Y and valuable data x and $f(x)$. First of all, every participant gets the same set

$$C = \{\text{hash}(y), c_1, c_2, \ldots, c_n\},$$

even if participants are in collusion with each other, they also cannot get more information from C. Next, when f is the one-to-one function, the preimages of

$$\{f(x_1), f(x_2), \ldots, f(x_n), y\}$$

have only one, respectively, $x_1, x_2, \ldots, x_n, x$. The participant is unable to distinguish between valuable x and pseudo valuable data $x_1, x_2, \ldots, x_n$. But when all

participants are in collusion, they can know whether the valuable data exists. We can improve the security of this scheme. In step 3, change the original criterion to

$$C = \{\text{hash}(y), c_1, c_2, \ldots, c_n, c_1^{'}, c_2^{'}, \ldots, c_t^{'}\},$$

in which, $c_1^{'}, c_2^{'}, \ldots, c_t^{'}$ are some random numbers that supervisor adds. If the valuable data exists, hash(y) and $c_1, c_2, \ldots, c_t$ will have only one preimage, respectively; if the rare valuable data does not exist, hash(y) and $c_1^{'}, c_2^{'}, \ldots, c_t^{'}$ will have not any preimage. As long as n and t are privately owned by the supervisor, every participant cannot determine whether the valuable data exists.

When f is a more-to-one function, if all participants are in collusion with each other, participants can find the hash(y) based on the number of preimages of c_i and hash(y). Usually, every c_i has a lot of preimages in X. However, the preimages of hash(y) are rare. Participants will find criterion hash(y) and valuable data. This scheme cannot resist the second and third deceive models.

6.2 Scheme II (Criterion Reduction)

1. The supervisor computes $h(y)$, and let $\hat{y}$ be the first k bits of the result, where k is a security parameter. The supervisor sends $\hat{y}$ to participants along with the task assignments.
2. For each assigned input x, a participant computes $f(x)$, and checks whether the first k bits of $h(f(x))$ equal $\hat{y}$. If true, the participant returns x and $h(f(x))$ to the supervisor; otherwise, discards x.
3. The supervisor verifies whether $h(f(x)) = h(y)$. If false, x is just ringer; else, x is a rare event.

Apparently, in the single compression method there exist larger defects. In the b) of scheme 2, participant P_i may only compute the part of the data in D_i, but the supervisor cannot detect this kind of deception. It can easily cause the loss of rare event.

By the above analysis, when the task function f is one-to-one function in the scheme I, it has a good security and can resist the three deceptive models. But when f is a more-to-one function, participants will be able to find the rare event that the supervisor is looking for. So the scheme I cannot resist the second and third deception models. The scheme II lacks the necessary honesty test steps, and even cannot determine whether participant do the honest computation for all the data in the task. Thus, it also has a lot of defects on security.

6.3 Three New Schemes

In this section, we propose three new schemes.

1) New Scheme I.

Firstly, we introduce Monte Carlo simulation [9, 16]. It is a technique that employs random numbers to solve problems in which time plays no substantive role. The technique involves simulating a random experiment a large number, say N, of times and recording the number of times, say C, that an event of interest occurs. The law of large numbers asserts that if N is large, the ratio C/N should be a good point estimate of the probability of the event occurring.

As a simple example, consider the problem of finding the area of a region S contained in the square $U \equiv [0,1] \times [0,1]$ in the xy-plane. Using Monte Carlo simulation, one can choose N points from a uniform distribution in U, and count the number of points, C, that lie in S. The approximation for the area would then be C/N.

This example is not well suited for a large scale distributed computation, but serves as an illustration of how the seeding technique can be applied to Monte Carlo simulations in general. The supervisor chooses a particular implementation for the random number generator (ensuring portability) and some number k of seeds. Before any tasks are assigned, an initial run of N/k replications is computed using one of the seeds s' chosen arbitrarily. This seed becomes the ringer for the remaining task assignments. Participants are then sent the code for the generator along with k seeds (including s'), and are instructed to run N/k replications with each of the seeds, returning the area estimate corresponding to each seed. An adversary cannot determine which of the k seeds is the ringer, and therefore cannot return results for fewer than k seeds without raising suspicion. The returned results can be checked for validity using the initial run generated with s'. In effect, the supervisor has managed to provide a measure of assurance while performing only $1/k$ of the work.

By their very nature, Monte Carlo simulations provide a form of redundancy because, provided the number of replications is sufficiently large, each task should return an estimate similar to the other tasks. However, seeding as described here augments the redundancy by enhancing the resistance to collusion.

We can use Monte Carlo simulations to verify whether the participant is honest. This scheme is available for one to one function and more to one function. At the same time, it can resist the semi-honest cheater and hoarding cheater.

Let n be the number of participants in the scheme. P_i $(i = 1, 2, \ldots, n)$ is a participant, and D_i is P_i' task set. Obviously $X = D_1 \cup D_2 \cup \ldots D_n$.

1. The supervisor randomly selects n inputs x_1, x_2, ..., x_n from the input domain X.
2. The supervisor computes $c_i = \text{hash}(f(x_i))$, for $i = 1, \ldots, n$ and $\text{hash}(y)$. Let $\hat{y}$ be the first k bits of $\text{hash}(y)$.
3. The supervisor randomly selects $D_i' \subset D_i$. For each input $x \in D_i'$, the supervisor computes $f(x)$, and checks whether the first k bits of $h(f(x))$

equal $\hat{y}$. If true, the supervisor adds x in $D_i^{''}$. Let p_i be the number $\frac{|D_i^{''}|}{|D_i^{'}|}$, if $p_i < \varepsilon$ (ε is a parameter), we select k again and repeat b). Otherwise, the supervisor can estimate the number of which satisfies that the first k bits of $h(f(x)), x \in D_i$ equal $\hat{y}$ is $p_i|D_i^{'}|$.

4. The supervisor sends $\hat{y}$ and $C = \{c_1, c_2, \ldots, c_n\}$ to participants along with the task assignments.
5. For each assigned input x, participant P_i computes $f(x)$, and checks whether the first k bits of $h(f(x))$ equal $\hat{y}$ or $h(f(x))$ equals C_i. If true, P_i returns x and $h(f(x))$ to the supervisor. Actually, the results which P_i returns are divided into two sets. One is $R_i = \{(x, h(f(x))) \mid x \in D_i$, the first k bits of $h(f(x))$ equal $\hat{y}\}$ and the other is

$$R_i^{'} = \{\{x \in D_i \mid \text{hash}(f(x)) = c_1\}, \ldots, \{x \in D_i \mid \text{hash}(f(x)) = c_n\}\}.$$

6. The supervisor checks whether $h(f(x))$ equals $h(y)$, $x \in R_i$. If true, x is the rare event. The supervisor also checks whether $\frac{|R_i|}{p_i|D_i^{'}|}$ approximately equals 1 and

$$x_i \in \{\{x \in D_i \mid h(If(x)) = c_i\}, \ldots, \{x \in D_i \mid h(f(x)) = c_n\}\}.$$

If true, P_i is a honest participant.

Security Analysis.

Because R_i contains more elements, p_i is unable to determine the real rare event. Hence whether for the one-to-one or more-to-one function, it can well hide the valuable data. Because $R_i^{'}$ must contain all the random inputs in D_i, P_i must compute all $f(x)$ for $x \in D_i$. So this scheme can resist semi-honest cheater and hoarding cheater. And at the same time, the validation of equation $\frac{|R_i|}{p_i|D_i^{'}|} \approx 1$ ensures the small possibility of missing rare events. On average, the feasibility and the security of the scheme are both relatively high.

2) New Scheme II.

Like most of schemes of grid computing, the above scheme is unable to resist malicious cheater. Because the supervisor did not verify the elements in R_i which were returned by participant P_i in order to destroy the supervisor's work, malicious cheater P_i obtains R_i, but he/she does not return the real R_i to the supervisor. So he/she constructs a new set $R_i^{'}$ in which the first k bits of every element equals $\hat{y}$, and $|R_i^{'}| = |R_i|$. In this way, the supervisor cannot find the cheat of P_i and loses some high-value rare events. For example, x_0 is a high-value rare element which is obtained by malicious cheater P_i, i.e., $f(x) = y, x_0 \in D_i$. P_i uses (x_0, A) to replace $(x_0, \text{hash}(x_0))$ (the first k bits of A equal $\hat{y}$ and

$A \neq \text{hash}(x_0)$. According to $A \neq \text{hash}(y)$, the supervisor thinks that x_0 is not a value element, so the high-value element will be lost.

In the following we present a simple improved scheme. Using the sample test of R_i, it can resist the three deception models, and the security is enhanced.

Let n be the number of participants in the scheme. P_i $(i = 1, 2, \ldots, n)$ is a participant, and D_i is the task of set of P_i, and

$$X = D_1 \cup D_2 \cup \ldots \cup D_n.$$

1. The supervisor randomly selects n inputs $x_1, x_2, \ldots, x_n$ from the input domain X.
2. The supervisor computes $c_i = \text{hash}(f(x_i))$, for $i = 1, \ldots, n$ and $\text{hash}(y)$. Let $\hat{y}$ be the first k bits of $\text{hash}(y)$.
3. The supervisor randomly selects $D_i^{'} \subset D_i$. For each input $x \in D_i^{'}$, the supervisor computes $f(x)$, and checks whether the first k bits of $h(f(x))$ equal $\hat{y}$. If true, the supervisor adds x in $D_i^{''}$. Let p_i be the number $\frac{|D_i^{''}|}{|D_i^{'}|}$, if $p_i < \varepsilon$ (ε is a parameter), we select k again and repeat b). Otherwise, the supervisor can estimate the number of which satisfies that the first k bits of $h(f(x)), x \in D_i$ equal $\hat{y}$ is $p_i|D_i^{'}|$.
4. The supervisor sends $\hat{y}$ and $C = \{c_1, c_2, \ldots, c_n\}$ to participants along with the task assignments.
5. For each assigned input x, participant P_i computes $f(x)$, and checks whether the first k bits of $h(f(x))$ equal $\hat{y}$ or $h(f(x))$ equals C_i. If true, P_i returns x and $h(f(x))$ to the supervisor. Actually, the results which P_i returns are divided into two sets. One is $R_i = \{(x, h(f(x))) \mid x \in D_i$, the first k bits of $h(f(x))$ equal $\hat{y}\}$ and the other is

$$R_i^{'} = \{\{x \in D_i \mid \text{hash}(f(x)) = c_1\}, \ldots, \{x \in D_i \mid \text{hash}(f(x)) = c_n\}\}.$$

6. The supervisor checks whether $h(f(x))$ equals $h(y)$, $x \in R_i$. If true, x is the rare event. The supervisor also checks whether $\frac{|R_i|}{p_i|D_i^{'}|}$ approximately equals 1 and

$$x_i \in \{\{x \in D_i \mid h(f(x)) = c_i\}, \ldots, \{x \in D_i \mid h(f(x)) = c_n\}\}.$$

   If true, P_i is a honest participant.
   If true, the supervisor will carry out sample test.
7. The supervisor randomly chooses m elements from R_i. Let

$$S = \{(s_1, A_1), (s_2, A_2), \ldots, (s_m, A_m)\}$$

be the set which is constructed by m elements and their computing values. The supervisor verifies whether $h(f(s_i))$ equals A_i for $i = 1, \ldots, m$. If true, P_i is honest. So the supervisor checks whether $f(f(x))$ equals $h(y)$, $x \in R_i$. If true, x is the rare event.

Theorem 1. When s samples are sampled by the supervisor in this scheme, the probability that a participant with honesty return ratio h can cheat successfully is

$$Pr(\text{cheating succeeds}) = h^s.$$

Proof. Since the sample number is very big, it is an independent and identically distributed probability. When the probability of one correct sample is h, the probability of s correct samples is h^s. □

Therefore, the probability that the supervisor finds the cheat of a malicious cheater P_i is $1 - h^s$.

To keep the probability of unsuccessful cheating above a big threshold ε, the sample size s should be

$$s > \frac{\log(1-\varepsilon)}{\log h}.$$

Table 1 shows how large s should be for different honesty return ratio h, given $\varepsilon = 0.9$ or 0.99. In fact, if $s > 500$, the probability that the supervisor finds the cheat of malicious cheater P_i is greater than 0.99.

h	$t = 0.9$	$t = 0.99$
0.5	4	7
0.8	11	21
0.9	22	44
0.99	230	459

Table 1.

3) New Scheme III.

In most of cheating detection schemes, the computation task X had not been dealt with. In order to achieve the purpose of hiding Y, they only used new criterion Y' to replace Y.

We present a new method. Let Y be $\{y\}$, we can construct a new set C, for $\forall x \in C$, $f(x) = y$. In this way, we expand the task to $X \cup C$.

The new scheme is described in the following:

1. The supervisor constructs a new collection C, for $\forall x \in C$, $f(x) = y$, and $C = C_1 \cup C_2 \cup \ldots \cup C_n$.

2. The supervisor computes hash(y), and let $\hat{y}$ be the first k bits, where k is a safety parameter. The supervisor sends $\hat{y}$ to participant P_i along with the task $D_i \cup C_i$, $i = 1, \ldots, n$.
3. For each assigned input x, participant P_i computes $f(x)$, and checks whether the first k bits of $h(f(x))$ equal $\hat{y}$. If true, P_i returns x and $h(f(x))$ to the supervisor.

Let $R_i = \{(x, h(f(x))) \mid x \in D_i \cup C_i$, the first k bits of $h(f(x))$ equal $\hat{y}\}$.

- The supervisor checks whether C_i is contained in R_i. If true, P_i is honest.
- The supervisor checks whether $h(f(x))$ equals $h(y)$. If true, x is the rare event.

If C_i is contained in R_i, P_i is honest. So P_i must compute every $f(x)$, $x \in D_i \cup C_i$. If not, the supervisor will find his dishonesty. Due to a participant cannot distinguish the data from D_i and C_i, the rare events can be disguised. In conclusion, the scheme can resist all three cheat modes and a collusion attack.

7 CONCLUSION

In this article, according to the computing power of the supervisor, we propose the deceptive detection schemes under two different circumstances, and combine the characteristics of the task function f to analyze the security of the deceptive detection. Based on the technology of double check, we proposed an improved scheme at the sacrifice of time, i.e. the secondary allocation scheme of double check. We reinforced the security of double check greatly. Finally, we analyzed the common problem of High-Value Rare Events, improved the deceptive detection scheme due to Du and Goodrich [1], and then put forward a new deceptive detection scheme with better security and efficiency.

REFERENCES

[1] Du, W.—Goodrich, M. T.: Searching for High-Value Rare Events with Uncheatable Grid Computing. Proceedings of the 3rd Applied Cryptography and Network Security Conference (ACNS), New York, NY, USA, June 2005, pp. 122–137, doi: 10.1007/11496137_9.

[2] Yu, J.—Wang, X.: Deceptive Detection and Security Reinforcement in Grid Computing. 2013 5th International Conference on Intelligent Networking and Collaborative Systems, 2013, pp. 146–152, doi: 10.1109/INCoS.2013.30.

[3] Climate Prediction Web Site. Available at: http://www.climateprediction.net.

[4] Distributed.net Web Site. Available at: http://www.distributed.net.

[5] IBM Grid Computing. Available at: http://www-1.ibm.com/grid/aboutgrid/what-is.shtml.

[6] SETI@Home. Available at: `http://setiathome.berkeley.edu`.

[7] The Smallpox Research Grid. Available at: `http://www-3.ibm.com/solutions/lifesciences/research/smallpox`.

[8] The Great Internet Mersenne Prime Search. Available at: `http://www.mersenne.org/prime.htm`.

[9] Search for Extraterrestrials' or Extra Cash. Available at: `http://www.dallasnews.com/technology/1202ptech9pcs.htm`.

[10] Law, A. M.—Kelton, W. D.: Simulation Modeling and Analysis. McGraw-Hill, 3rd edition, 2000.

[11] Aiello, W.—Bhatt, S.—Ostrovsky, R.—Rajagopalan, S. R.: Fast Verification of Any Remote Procedure Call: Short Witness-Indistinguishable One-Round Proofs for NP. In: Montanari, U., Rolim, J. D. P., Welzl, E. (Eds.): Automata, Languages and Programming (ICALP 2000). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1853, 2000, pp. 463–474, doi: 10.1007/3-540-45022-x_39.

[12] Anderson, D. P.: BOINC: A System for Public-Resource Computing and Storage. Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Washington, DC, USA, November 2004, pp. 4–10, doi: 10.1109/grid.2004.14.

[13] Beigel, R.—Margulis, G.—Spielman, D. A.: Fault Diagnosis in a Small Constant Number of Parallel Testing Rounds. Proceedings of the Fifth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA '93), 1993, pp. 21–29, doi: 10.1145/165231.165234.

[14] Brin, S.—Page, L.: The Anatomy of Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, Vol. 30, 1998, No. 1–7, pp. 107–117, doi: 10.1016/s0169-7552(98)00110-x.

[15] Golle, P.—Mironov, I.: Uncheatable Distributed Computations. In: Naccache, D. (Ed.): Topics in Cryptology (CT-RSA 2001). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2020, 2001, pp. 425–440, doi: 10.1007/3-540-45353-9_31.

[16] Golle, P.—Stubblebine, S. G.: Secure Distributed Computing in a Commercial Environment. Proceedings of the 5th International Conference on Financial Cryptography, British West Indies, February 2001, pp. 289–304, doi: 10.1007/3-540-46088-8_23.

[17] Szajda, D.—Lawson, B.—Owen, J.: Hardening Functions for Large Scale Distributed Computations. Proceedings 2003 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, May 2003, pp. 216–224, doi: 10.1109/SECPRI.2003.1199338.

[18] Du, W.—Jia, J.—Mangal, M.—Murugesan, M.: Uncheatable Grid Computing. Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS), Hachioji, Tokyo, Japan, March 2004, pp. 4–11, doi: 10.1109/icdcs.2004.1281562.

[19] Minsky, Y.—van Renesse, R.—Schneider, F. B.—Stoller, S. D.: Cryptographic Support for Fault-Tolerant Distributed Computing. Proceedings of the Seventh ACM SIGOPS European Workshop: System Support for Worldwide Applications (EW 7), Connemara, Ireland, September 1996, pp. 109–114, doi: 10.1145/504450.504472.

[20] Kahney, L.: Cheaters Bow to Peer Pressure. Wired Magazine, February 15, 2001. Available at: https://www.wired.com/2001/02/cheaters-bow-to-peer-pressure/.

[21] Kuhn, M.—Schmid, S.—Wattenhofer, R.: Distributed Asymmetric Verification in Computational Grids. Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, Florida USA, April 2008, pp. 1–10, doi: 10.1109/ipdps.2008.4536244.

Jianhua Yu is Ph.D. student at the South China Normal University. He is Lecturer at the School of Mathematical Sciences, South China Normal University. His research interests include issues related to cryptography, computer network, and mathematical modeling. He is author of a great deal of research studies published in national and international journals and conference proceedings.

Yuan Li received his Ph.D. in number theory at the State University of New York. He is Associate Professor at the Department of Mathematics, Winston-Salem State University. His research interests include computational number theory and cryptography.