Toward an Optimal Redundancy Strategy for Distributed Computations*

Doug Szajda Barry Lawson Jason Owen
University of Richmond
Richmond, Virginia
{dszajda, blawson, wowen}@richmond.edu

Abstract

Volunteer distributed computations utilize spare processor cycles of personal computers that are connected to the Internet. The related computation integrity concerns are commonly addressed by assigning tasks redundantly. Aside from the additional computational costs, a significant disadvantage of redundancy is its vulnerability to colluding adversaries. This paper presents a tunable redundancy-based task distribution strategy that increases resistance to collusion while significantly decreasing the associated computational costs. Specifically, our strategy guarantees a desired cheating detection probability regardless of the number of copies of a specific task controlled by the adversary. Though not the first distribution scheme with these properties, the proposed method improves upon existing strategies in that it requires fewer computational resources. More importantly, the strategy provides a practical lower bound for the number of redundantly assigned tasks required to achieve a given detection probability.

Keywords: distributed computation, probabilistic verification

1. Introduction

Distributed volunteer computing platforms, in which personal computers connected to the Internet volunteer idle processor cycles, facilitate computations that were once unobtainable without the use of a large cluster or supercomputer. Typically, the *supervisor* of a volunteer distributed computing platform recruits *participants* who agree to allow the supervisor to execute code on their personal computers. Participants then download code that serves as the local execution environment for assigned computational *tasks*, which are typically small enough to be handled in a few hours by a typical personal computer. For a given computation, participants are chosen, tasks are assigned and transmitted, and, as tasks are completed, significant results are collected by the supervisor. Though task results may be related, the tasks themselves are independent, so no communication is necessary between participants. Examples of current endeavors include the Folding@home and SETI@home projects and Apple's XGrid.

Because code is executed in untrusted environments, several security concerns are raised. These include the potential for participants to intentionally or unintentionally corrupt results and for participants to claim credit for work not completed. The standard approach to hardening these computations is simple redundancy, in which k (typically 2) copies of each task are distributed. While providing a measure of protection, simple redundancy is vulnerable to collusion since matching returned results are typically assumed to be correct. Thus in the absence of additional verification mechanisms (which are typically very expensive), an adversary controlling two copies of the same task can cheat with impunity. Moreover, obtaining multiple copies of some tasks is relatively easy. A dedicated individual can obtain hundreds of user names, each of which can be assigned thousands of tasks[1]. An obvious variation is to assign tasks redundantly, but to require that only a single copy of a given task is outstanding at any time. This doubles both the resource and time costs of the computation, but still does not eliminate the collusion threat — in an N task computation, an adversary who controls at least $1/\sqrt{N}$ proportion of the participants is expected to be able to cheat on at least one task without being caught (justification in Appendix A).

Golle and Stubblebine [3], in the course of describing a general security based administrative framework for commercial distributed computations, present a task distribution scheme that offers significant improvement over simple redundancy. Their strategy relies on probabilis-

*This work was partially supported by the National Science Foundation under grant IIS-0524239.

[1]The Search for Extra-Terrestrial Intelligence project [12] conducted by SETI@Home, for example, has experienced days in which more than 5000 new user names were assigned, and boasts participants who have *averaged* more than 1000 tasks completed each day since the project began.

tic redundant verification to guarantee increased protection against collusion by ensuring that (in theory) adversaries face detection probabilities above a predetermined threshold – regardless of the number of copies of a task the adversary controls. Moreover, in many cases their scheme provides significant resource savings over simple redundancy. Their work did not consider the modifications required to apply their approach to a real world implementation.

Relative to the work of Golle and Stubblebine [3], we have considered additional schemes that provide equal protection while requiring fewer computational resources. This leads naturally to the primary question considered in this paper: *Does there exist a distribution scheme that uses the minimum possible resources yet guarantees a given probability of detection, regardless of the number of copies of a task the adversary controls?*

We present here a distribution, the *Balanced distribution*, which achieves an optimal *practical* use of computing resources. We emphasize the "practical" modifier for several reasons. First, implicit in the designation "optimal" is a well-defined mathematical problem. In the present context, the natural mathematical model is a linear programming problem in which constraints are dictated primarily by conditions on the probability of detecting cheaters. These constraints, as we will show later, cannot be met in a finite dimensional system unless the supervisor is required to precompute some tasks. Reducing the number of precomputed tasks, however, requires increasing the number of variables in the linear system. This results in greatly decreased cheat detection probabilities for certain tasks when the adversary controls a nontrivial number of participants. Thus distributions that are optimal in that they require the minimum amount of computing resources are not necessarily the distributions that best satisfy the protection requirements of these computations.

The Balanced distribution, on the other hand, requires a trivial amount of precomputing, and is robust in the presence of an adversary controlling a large proportion of participants. Moreover, when this last consideration about the adversary is included in the relevant linear programming model, the Balanced distribution is the resulting optimal solution. Thus the Balanced distribution represents the best balance of efficient use of computing resources, low precomputing requirements, and relatively good resistance to an adversary controlling a significant number of participants.

The specific contributions of this paper are the following:

- Provide a theoretical lower bound for the amount of redundancy required to guarantee a given cheating detection threshold.
- Present the Balanced distribution, a theoretical task distribution scheme that requires significantly fewer resources than simple redundancy while improving protection from colluding adversaries controlling a significant number of participants.
- Present techniques that allow this theoretical distribution to be applied to real world systems.

Finally, two caveats. As with any probability based scheme, security is based on providing an environment in which the *likelihood* of detecting malicious activity is increased. Because there is no *guarantee* that malicious activity will be detected, a determined adversary will succeed in disrupting the system if she makes a sufficient number of attempts to do so. It is highly likely, however, that in making these attempts she will be detected, alerting the supervisor to the presence of an active adversary, and allowing for potential (though likely more expensive) reactive measures.

Second, we make no claim that our solution, or any static redundancy based solution, provides the "best" solution to the problem of guaranteeing the integrity of distributed computations. There are, for example, several alternative schemes proposed for improving security in these computations (see e.g., [2, 6, 10, 13]), many of which do not rely on static redundancy. The fact remains, however, that all current implementations of which we are aware rely on static redundancy schemes (typically simple redundancy) for protecting computation integrity. It is thus prudent to consider whether there exist better, or even optimal, static redundancy schemes.

We argue that the Balanced distribution is the best overall distribution scheme (when all factors are considered) for three key reasons:

1. It provides significantly better protection against collusion than simple redundancy, often with decreased computing costs.
2. It requires fewer compute resources than any currently proposed static redundancy scheme and a negligible amount of precomputing.
3. Perhaps most important, because ours is a static redundancy scheme, deployment costs are trivial: current implementations can be easily modified to execute the strategy.

The remainder of the paper is organized as follows. In Section 2 we briefly discuss our model of the distributed computations and platforms under consideration, and provide notation and terminology. Sections 3 and 4 motivate and present the Balanced distribution. Section 5 considers the effect of an adversary controlling a significant number of participants. Sections 6 and 7 discuss the modifications required for implementation in real systems, as well as an

extension of the Balanced distribution. Section 8 covers related work and conclusions are presented in Section 9.

2. The model

We assume the same terminology and model as described in [13]. To further clarify, we use the term *assignment* to denote the code and data allocated to a participant. Assignments are not distinct — there can, and often will, be multiple identical assignments in a computation. Technically, we consider tasks to be members of a set, while assignments are members of a multiset whose elements are tasks. If there are exactly k copies of a task in a computation, then we say that the task has *multiplicity* k or equivalently is a *multiplicity* k task.

We assume the existence of a global intelligent adversary controlling multiple participants. The adversary has knowledge both of the algorithm used for the computation and of the measures used to protect the integrity of the computation. Implicit in the assumption of a global adversary is that collusion is both possible and likely, and that the adversary has no knowledge of tasks not assigned to her. When the adversary has been assigned k copies of a given task, we say that she *controls* k copies of the task. An adversary *cheats* on a task by attempting to return an identical incorrect result on all copies of the task in her possession.

Finally, attacks that result from a compromise of data in transit are beyond the scope of this paper — we assume the integrity of such data is verified by other means. In addition, we do not consider attacks that result from the compromise of the central server or other trusted management nodes.

2.1. Terminology and Notation

A redundancy-based distribution scheme for a distributed computation is specified by the number of tasks that are to be assigned with multiplicity i, for $i = 1, 2, 3, \ldots$. We assume for the remainder of the paper that the computation under consideration consists of N tasks (though certainly more than N assignments). We use vector notation to describe specific distributions. That is, we use the vector $\bar{x} = (x_1, x_2, x_3, \ldots)$, with nonnegative components to describe the distribution in which exactly x_i tasks have multiplicity i. Equivalently, the proportion of tasks assigned with multiplicity i is x_i/N. For example, simple redundancy, in which all tasks are assigned exactly twice, would correspond to the vector $(0, N, 0, 0, \ldots)$, and in this example the proportion of tasks assigned with multiplicity 2 is $x_2/N = 1$.

In any real implementation, there will be an upper bound on task multiplicities. Rather than impose an a priori upper limit on multiplicities, for theoretical reasons we allow distributions with arbitrarily large multiplicities. We define the *dimension* of distribution $\bar{x} = (x_1, x_2, x_3, \ldots)$ to be the largest index i for which x_i is nonzero, if such an i exists, and to be infinite otherwise. Since each of the N tasks must be assigned a multiplicity, we must have $\sum_{i=1}^{\infty} x_i = N$, and as a result that $\lim_{n\to\infty} x_n = 0$. Finally, the total number of assignments in a distribution is given by $\sum_{i=1}^{\infty} i x_i$. We define the *redundancy factor* of a distribution to be the ratio $(\sum_{i=1}^{\infty} i x_i)/N$. As an example, note that for simple redundancy the redundancy factor is $2x_2/N = 2N/N = 2$.

Implicit in our acceptance of infinite dimensional distributions is the potential for the assignment of arbitrarily small fractions of tasks (i.e., that x_i can be real valued but nonnegative). Just as infinite dimensional distributions are not possible in real systems, so too are such fractions of tasks not possible. However, allowing them in a theoretical context simplifies the analysis. These theoretical assumptions are easily modified for actual implementations, as we describe in Section 6.

2.2. Constraints

We consider distributions in which the probability of detecting an adversary who cheats is no smaller than some fixed level α, with $0 < \alpha < 1$. Toward this, let $P_{k,p}$ denote the probability that an adversary who controls proportion p of the total assignments is detected when cheating on a task of which she controls k copies. We let P_k denote the probability that an adversary who controls exactly k assignments, all of them copies of the same task, is detected when attempting to cheat on that task. This *asymptotic* detection probability satisfies $P_k = \lim_{p\to 0} P_{k,p}$, and thus in practical terms gives the probability that an adversary is detected when she controls a small proportion of participants. For clarity we focus our attention for the moment on the asymptotic probability.

For k a positive integer, define a k-tuple to be k copies of the same task. Then for a distribution $\bar{x}$, we have

$$\begin{aligned} P_k &= \frac{\text{\# of } k\text{-tuples from tasks assigned} > k \text{ times}}{\text{\# of } k\text{-tuples from tasks assigned} \geq k \text{ times}} \\ &= \frac{\sum_{i=k+1}^{\infty} \binom{i}{k} x_i}{x_k + \sum_{i=k+1}^{\infty} \binom{i}{k} x_i}. \end{aligned}$$

Therefore we have the following equivalent inequalities:

$$\begin{aligned} & P_k \geq \alpha \\ \iff & \alpha \left(x_k + \sum_{i=k+1}^{\infty} \binom{i}{k} x_i \right) \leq \sum_{i=k+1}^{\infty} \binom{i}{k} x_i \\ \iff & x_k \leq \left(\frac{1-\alpha}{\alpha} \right) \sum_{i=k+1}^{\infty} \binom{i}{k} x_i. \end{aligned}$$

For k a positive integer, we let C_k denote the constraint that $P_k \geq \alpha$ (i.e., the probability of detecting an adversary who controls k copies of a task is greater than or equal to α). We let C_0 denote the constraint that $\sum_{i=1}^{\infty} x_i = N$.

We say a distribution $\bar{x}$ represents a *valid infinite dimensional distribution* if $\sum_{i=1}^{\infty} x_i = N$, each x_i is nonnegative, and $P_k \geq \alpha$ for all positive integers k. The vector $\bar{x}$ represents a *valid m-dimensional distribution* if $x_i = 0$ for $i > m$, $\sum_{i=1}^{m} x_i = N$, each x_i is nonnegative, and $P_k \geq \alpha$ for $k = 1, 2, \ldots, m-1$. That is, infinite dimensional distributions are valid if they assign each of the N tasks a non-negative multiplicity, and satisfy all detection probability constraints. Finite dimensional distributions, on the other hand, are valid if they assign each of the N tasks a non-negative multiplicity, and satisfy all but one of the detection probability constraints. Specifically, the constraint that $P_m \geq \alpha$ cannot be satisfied by a valid m-dimensional distribution, since in the absence of additional security mechanisms (such as the supervisor verifying all multiplicity m tasks), an adversary who controls m copies of a task can cheat without detection.

These observations have interesting implications for real systems. Since clearly any implementation will have an upper bound on task multiplicities, only finite dimensional distributions can be implemented. Thus in order to meet all probability constraints, the supervisor must verify some tasks. We assume then for the remainder of the article that all valid finite dimensional distributions are augmented by some form of verification that ensures that *all* probability constraints are met. Exactly which and how many tasks must be verified is a question that will be considered further when the issue of optimal distributions is discussed in Section 4. For now, we impose only one loosely defined constraint: the amount of precomputing must be reasonably small. In particular, the distribution in which all tasks are precomputed trivially satisfies all of the required probability constraints, but is of little practical use, and we thus do not consider it or any solutions that require similarly large amounts of precomputing.

3. Motivation

3.1. The Golle-Stubblebine Distribution

The first (and to our knowledge only, prior to the current article) distribution scheme that meets the required probability constraints for all positive integers k is that of Golle and Stubblebine [3]. They propose the assignment $\bar{g} = \{g_i\}$ defined for $i = 1, 2, \ldots$ by $g_i = (1-c)c^{i-1}N$, for fixed c with $0 < c < 1$. Straightforward analysis shows that $\sum_{i=1}^{\infty} g_i = N$, and that the total number of assignments required is $N/(1-c)$, or equivalently that the redundancy factor is $1/(1-c)$. They show that the asymptotic probability that the adversary is detected while cheating on a k-tuple is

$$P_k = 1 - (1-c)^{k+1}.$$

More generally,

$$P_{k,p} = 1 - (1 - c(1-p))^{k+1}.$$

Note that these probabilities increase with increasing k. In order to guarantee that the detection probability remains at or above level α, it suffices to guarantee that $P_{1,p} \geq \alpha$. In the asymptotic case, this requires $\alpha = 1-(1-c)^2$ or equivalently $c = 1 - \sqrt{1-\alpha}$. The redundancy factor associated with detection probability α in the Golle-Stubblebine distribution is thus $1/\sqrt{1-\alpha}$, so that in the asymptotic case, their scheme requires fewer resources than simple redundancy provided $\alpha < 0.75$. The corresponding non-asymptotic redundancy factor is $\frac{1-p}{\sqrt{1-\alpha}-p}$.

It was the observation that detection probabilities increase with increasing k that suggested the potential for a more efficient (and possibly provably *most* efficient) distribution of tasks. An adversary with knowledge of the Golle-Stubblebine distribution scheme knows that she encounters the most favorable odds of cheating without detection when she cheats on a task of which she possesses a single copy. Thus the apparent additional protection against cheating on k-tuples with $k > 1$ is illusory, since an intelligent adversary will not attempt to cheat on these. The consequence is that the "extra" assignments required by this scheme are wasted resources. Regardless, this distribution is an elegant solution to the problem for two reasons: for appropriately chosen c values, (i) the adversary is faced with an unfavorable (from the adversary's perspective) probability of being detected regardless of the number of copies of a task she possesses, while (ii) the total assignment cost can be significantly less than simple redundancy.

3.2. The Assignment Minimizing Distributions

In moving toward a resource-optimal distribution, the concern is exclusively with minimizing the number of assignments required by a distribution. If one assumes that the adversary can obtain control of only a trivial proportion of assignments (an assumption we have argued is in general dangerous for these computations), then determining the best distribution involves a linear programming problem in either finite or infinite dimensions, in terms of the probability constraints C_i from Section 2.2. Specifically, let S denote the infinite dimensional system that seeks to minimize the function $M(x_1, x_2, x_3, \ldots) = \sum_{i=1}^{\infty} i x_i$ subject to the constraints that both the x_i are nonnegative, and C_i for $i \geq 0$ are satisfied. Let S_k denote this same system, with the exception that only the C_i for $i < k$ need be satisfied (in addition to the nonnegativity constraints). Note

that in S_k, x_k is effectively unconstrained (except that it must be nonnegative). The following proposition provides a lower bound on the total number of requirements necessary for the optimal solution to these systems. The proof is presented in Appendix B.

Proposition 1. An optimal solution to system S or to any S_k must require strictly greater than $\frac{2N}{2-\alpha}$ total assignments. Equivalently, the optimal redundancy factor is strictly greater than $\frac{2}{2-\alpha}$.

Because solving any of the finite dimensional systems S_k is an elementary linear programming problem, we present the following facts without proof. Though the solutions for all α exhibit similar behavior, their exact form varies with α. We present the $\alpha = 0.5$ solution as a representative model.

Fact. The optimal solution to S_k for $k \geq 6$ is the distribution $\bar{x}$ with

$$x_1 = \frac{2Nk^2}{2-k+3k^2}, \qquad x_2 = \frac{Nk(k-1)}{2-k+3k^2}, \qquad x_k = \frac{2N}{2-k+3k^2} \quad (1)$$

and $x_i = 0$ for $3 \leq i \leq k-1$. The redundancy factor for this distribution is $4k^2/(2-k+3k^2)$.

Optimal solutions for other α values share the similar characteristics: most tasks are distributed either once or twice, but there is a small tail portion with just enough mass to ensure that constraints C_1 and C_2 are satisfied. Sometimes the tail mass is concentrated only on x_k; sometimes there is also mass at x_{k-1}. All other components of $\bar{x}$ are zero.

Note that as k grows large, the redundancy factor for the finite dimensional solution approaches the lower bound redundancy factor of $4/3$ described in Proposition 1 (with $\alpha = 0.5$). This is the typical behavior of the optimal solutions to the S_k.

We mentioned above that these distributions are optimal if one considers number of assignments exclusively and assumes that the adversary controls relatively few assignments. In practical systems, two additional important factors play a role: 1) the amount of precomputing required; and 2) non-asymptotic probabilities (i.e., assuming the adversary can control a nontrivial proportion of assignments). We consider the issue of non-asymptotic probabilities in greater detail in Section 5. For now, we make three general observations regarding the behavior of the assignment minimizing distributions. First, as the dimension increases, the total number of assignments required for the optimal solution decreases[2]. Second, as the dimension increases, the amount of precomputing required by the optimal solution decreases. Finally, as the dimension increases, the detection probability for some assignments in the distribution decreases rapidly. There are localized exceptions to these rules. Consider, for example, the detection probabilities for the minimizing distributions for $\alpha = 0.5$ shown in Figures 1 and 2. In moving from the solution for S_5 to the solution for S_6, the amount of precomputing *increases* from 602 tasks to 1923 tasks. In addition, in moving from systems S_3 to S_4, the redundancy factor increases, again violating the general principles above. However, in all cases, the validity of the observations in terms of global behavior remain valid. Similar behavior is observed in these systems for all relevant α values: ensuring that precomputing levels are sufficiently low requires moving to higher dimensional systems. Systems with higher dimension, however, suffer from detection probabilities that decrease rapidly for some task multiplicities as we consider non-asymptotic probabilities (as depicted in the last few rows in Figure 2. The reason for this is that for the assignment minimizing distributions, the vast majority of tasks are distributed once or twice (with a small amount distributed at the tail). As the adversary gains control of even a small proportion of tasks, she will obtain more than two copies of some. The non-asymptotic probability then becomes a conditional probability, and the knowledge that some tasks under her control

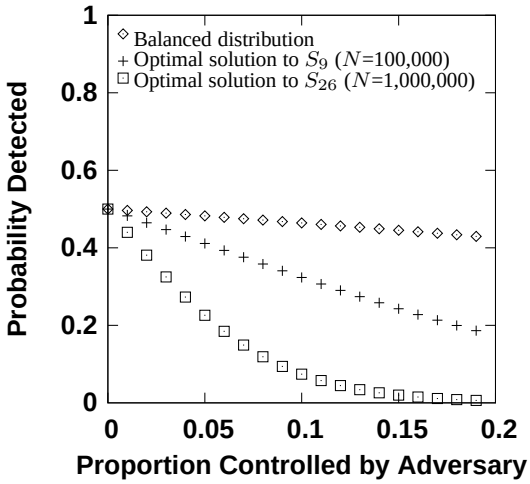

Figure 1. Detection probabilities for three different distributions ($\alpha = 1/2$ in each). Optimal solutions for S_9 and S_{26} are plotted. These represent (for the given N values) the first finite dimensional solutions that require fewer than 1000 precomputed tasks.

[2]This somewhat counterintuitive result stems from the special form of the optimal solutions: as in Equation (1), task multiplicities are either 1, 2, or k, with most tasks assigned once or twice. As k increases, fewer k multiplicity tasks are required to "cover" the tasks assigned with multiplicity one or two.

Dim.	Precomputing Required	Redundancy Factor	Min. P_k $p = 0.05$	Min. P_k $p = 0.1$	Min. P_k $p = 0.2$
1	100000	1.000	1.000	1.000	1.000
2	33333	1.333	0.487	0.474	0.444
3	7692	1.384	0.487	0.474	0.444
4	1333	1.387	0.487	0.474	0.444
5	602	1.386	0.474	0.448	0.390
6	1923	1.385	0.449	0.396	0.291
7	1408	1.380	0.436	0.371	0.247
8	1075	1.376	0.424	0.347	0.208
9	848	1.373	0.411	0.324	0.173
10	685	1.370	0.399	0.301	0.144
15	302	1.359	0.339	0.203	0.052
20	169	1.354	0.284	0.131	0.018
25	108	1.350	0.235	0.081	0.006
30	75	1.347	0.192	0.05	0.002
Bal.	1	1.386	0.482	0.464	0.426

Figure 2. Comparing the amount of precomputing required, redundancy factor, and lowest detection probabilities for the assignment minimizing distributions of a given dimension. Parameter values are $N = 100,000$ and $\alpha = 0.5$. The final line gives figures for the Balanced distribution.

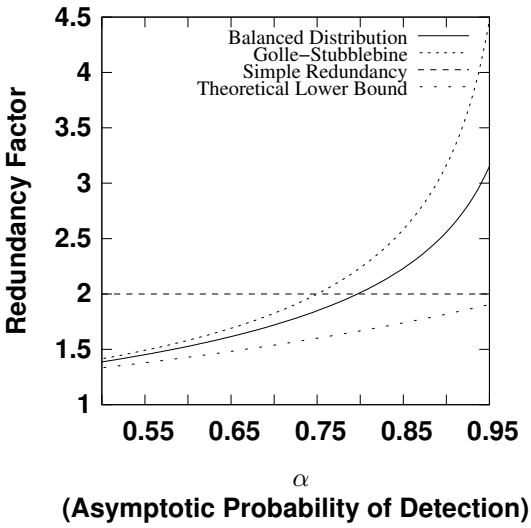

Figure 3. Redundancy factors for the Balanced and Golle-Stubblebine distributions. Included are the factors for simple redundancy and the theoretical minimum redundancy factor.

are in the tail of the distribution changes her odds of being caught on a two-tuple.

One final word on precomputing. We are reluctant to assign an arbitrary limit to the multiplicities of tasks, since the capacities of supervisors (and their willingness to perform verification) will vary. Regardless, it should be noted that precomputing a significant number of tasks is especially costly since in addition to the usual overhead of redundancy, computing requirements are effectively reassigned from the untrusted participants to the relatively few *trusted* nodes of the supervisor. This wastes resources that might be used for additional security measures or post-processing.

4. The Balanced Distribution

For fixed detection probability threshold α, let the *Balanced Distribution* $\bar{a} = \{a_i\}$ be defined for $i = 1, 2, \ldots$ by

$$a_i = N \left(\frac{1-\alpha}{\alpha}\right) \frac{1}{i!} \left(\ln\left(\frac{1}{1-\alpha}\right)\right)^i \qquad (2)$$

Theorem 1. For α with $0 < \alpha < 1$ the distribution $\bar{a}$ has the following properties:

1. $\sum_{i=1}^{\infty} a_i = N$.
2. $P_k = \alpha$ for all positive integers k.
3. The total number of assignments required is

$$\frac{N}{\alpha} \ln\left(\frac{1}{1-\alpha}\right).$$

The proof is presented in Appendix C. Figure 3 compares the redundancy factor of the Balanced distribution with both the Golle-Stubblebine distribution and simple redundancy. The redundancy factor of the Balanced distribution is less than that of the Golle-Stubblebine distribution for $0 < \alpha < 1$, and less than simple redundancy when $\alpha \leq 0.796812$. Figure 4 shows example Balanced and Golle-Stubblebine distributions for $N = 1,000,000$ and $\alpha = 0.75$. The final two non-zero entries in both the Balanced and Golle-Stubblebine distributions represent the tail modifications with ringers as presented in Section 6. Note that for these parameter values, the Balanced distribution provides a savings of more than 150,000 assignments over both the Golle-Stubblebine and simple redundancy distributions.

Though the Balanced distribution requires more total assignments than the Assignment Minimizing distributions, it has the important advantage that it requires a negligible amount of precomputing while at the same time remaining robust to collusion in the sense that detection probabilities decrease relatively slowly as the proportion p of assignments controlled by the adversary increases.

5. Non-Asymptotic Detection Probabilities

We consider in this section the issue of efficiency of a distribution in the non-asymptotic case (i.e., when the adversary controls a significant proportion of the assignments). Intuitively, if the non-asymptotic probabilities of a

Dim.	Balanced Distribution	Golle-Stubblebine	Simple Redundancy
1	462098	500000	0
2	320302	250000	1000000
3	148010	125000	0
4	51296	62500	0
5	14222	31250	0
6	3286	15625	0
7	650	7812	0
8	112	3906	0
9	17	1953	0
10	2	976	0
11	5	488	0
12	3	244	0
13	0	122	0
14	0	61	0
15	0	30	0
16	0	15	0
17	0	7	0
18	0	3	0
19	0	1	0
20	0	7	0
21	0	3	0
Tasks	1848452	2000174	2000000
Redund. Factor	1.848	2.0	2.0

Figure 4. Comparing task assignments for Balanced, Golle-Stubblebine, and simple redundancy. Parameter values are $N = 1,000,000$ and $\alpha = 0.75$. Figures for tail partition and ringers are included.

distribution vary with k then the distribution is inefficient. That is, the effective detection probability provided by a distribution is the minimum, over all relevant k, of $P_{k,p}$. When the $P_{k,p}$ values for a fixed p vary with k, then resources (extra copies of tasks) are being used to increase the detection probabilities for some values of k to levels above this effective detection level. These extra resources increase computation costs without increasing protection and are thus effectively wasted.

Given this, we consider in the next proposition the asymptotic properties a distribution must exhibit if that distribution is to be the most efficient (in terms of asymptotic total assignments) distribution that is *also* efficient in non-asymptotic cases. The proposition shows that such a distribution must exhibit equality in all relevant constraints C_i (i.e., $P_k = \alpha$).

Proposition 2. Let p be a fixed proportion, $0 \leq p < 1$, and fixed d be either a positive integer or ∞. Let $\mathcal{E}_d$ be the set of all valid distributions of dimension d such that $P_{k,p}$ is independent of k. Then the distribution in $\mathcal{E}_d$ that requires the fewest total assignments must have asymptotic detection probabilities P_k satisfying $P_k = \alpha$ for $1 \leq k < d$ (where in the case $d = \infty$, the property holds for all positive integers k). [Proof presented in Appendix D.]

In fact, this property is satisfied by the Balanced distribution, as the following proposition asserts. The proof is presented in Appendix E.

Proposition 3. Assume that tasks are assigned according to the Balanced distribution. Let p be the proportion of the total number of assignments controlled by the adversary. Then for k a positive integer, the probability that the adversary is caught while attempting to cheat on a k-tuple of identical assignments is

$$P_{k,p} = 1 - (1 - \alpha)^{1-p}.$$

Proposition 2 only asserts that a good distribution scheme must achieve equality in the C_i, not that achieving equality guarantees an optimal distribution. In practice, however, when the S_k systems are augmented so that the solution must satisfy that $P_k = \alpha$, the resulting optimal solutions are virtually indistinguishable from the Balanced distribution. We thus conclude that the Balanced distribution is the "best" distribution in that it provides the best balance of low redundancy factor, extremely low precomputing requirements, and efficiency in the non-asymptotic case.

6. Implementing the Strategy

The theoretical asymptotic analysis presented above ignores two important realities. First, at some point in the sequence $\{a_i\}$, the value of a_i will become less than one, and thus the scheme requires the assignment of a fractional number of tasks (an impossibility in practice). Second, in any real implementation there is an upper bound on the number of copies of any individual task (i.e., i is finite). A less important consideration is that each a_i value given by Equation (2) will almost always require rounding to handle the decimal values (i.e., so that in practice the number of tasks assigned a given multiplicity is an integer). All of these issues are easily handled as we show in the following suggested adaptation.

First, we round the size a_i of each partition down to the nearest integer. Next, we address the issue of a_i eventually falling below 1. Let i_f denote the first value of i for which $a_i < 1$. Assuming that α is bounded by 0.99 (we expect $\alpha < 0.8$ in practice), straightforward algebra shows that i_f is $O(\log((1-\alpha)N/\alpha))$. When a_i falls below 1, not all of the N tasks will have yet been assigned (in fact, only $\sum_{i=1}^{i_f-1} a_i$ of the tasks). The remaining tasks are assigned to one last partition, the *tail partition*, each with multiplicity i_f.

The tail partition is small — according to Lagrange's formula for the remainder of a Taylor series, $\sum_{i=i_f}^{\infty} a_i \leq 1/(1-\alpha)$ since $a_{i_f} < 1$. Including the effect of rounding down those a_i where $i < i_f$, the number of tasks assigned to the tail partition is no greater than $i_f + 1/(1-\alpha)$. This estimate is conservative: for the (extreme) values $N = 10^7$ and $\alpha = 0.99$, $i_f = 20$, and $a_{20} = 12$. The tail partition

size (including copies) is 240 tasks compared to 4,651,688 total assignments in the computation. Hence, the number of tasks (including copies) created by using a tail partition is a negligible fraction of the total number of tasks assigned. Moreover, the slight changes these modifications cause to detection probabilities are inconsequential, with the exception of the detection probability associated with multiplicity i_f tasks.

In practice, the adversary will not know the value of i_f and thus will not know that she can cheat on tasks of which she possesses i_f copies. Even if she can learn i_f, there is an easy solution to this problem: distribute $i_f + 1$ copies of a "ringer" task that has been precomputed by the supervisor. If r ringer tasks are precomputed, then the asymptotic probability that the adversary is detected when attempting to cheat on a task of which she possesses i_f copies is

$$1 - \frac{x_{i_f}}{x_{i_f} + \binom{i_f+1}{i_f} r} = 1 - \frac{x_{i_f}}{x_{i_f} + (i_f + 1)r}.$$

Equivalently, in order to ensure at least asymptotic probability α of detection in the i_f-tuple situation, one must choose r satisfying

$$r > \frac{\alpha x_{i_f}}{(1-\alpha)(i_f+1)}.$$

With the values from our example above ($x_{i_f} = 12$, $i_f = 20$, and $\alpha = 0.99$), at least 57 ringers are required. Again, these values are extreme. A more typical computation would have $N = 1,000,000$, and $\alpha = 0.75$, in which case $i_f = 11$ and $x_{i_f} = 5$. In this case, meeting level-α security requires only 2 ringers. Furthermore, the use of ringers increases the probability an adversary is caught for *all* values of i.

7. Extending the Main Theorem

Our distribution can be generalized to handle situations in which the supervisor wishes to guarantee a minimal redundancy level k for all tasks. The resulting distribution is defined for $i \geq k$ by

$$a_i = N\beta \frac{1}{i!}\left(\ln\left(\frac{1}{1-\alpha}\right)\right)^i$$

with

$$\beta = \frac{1}{\left(\frac{1}{1-\alpha}\right) - \sum_{i=0}^{k-1} \frac{1}{i!}\left(\ln\left(\frac{1}{1-\alpha}\right)\right)^i}.$$

As in the main theorem, the asymptotic detection probability for all $i \geq k$ is α. The redundancy factor is

$$\beta\left(\left(\frac{1}{1-\alpha}\right) - \sum_{i=0}^{k-2} \frac{1}{i!}\left(\ln\left(\frac{1}{1-\alpha}\right)\right)^i\right)\ln\left(\frac{1}{1-\alpha}\right).$$

For example, the redundancy factors for $\alpha = 0.5$ and distributions that require minimum multiplicities of 2,3,4, and 5, are respectively 2.259, 3.192, 4.152, and 5.126. Thus a supervisor using simple redundancy on a computation that has $N = 100,000$ tasks can guarantee an asymptotic cheat detection probability of 0.5 by assigning an additional 25,900 tasks (i.e., 13% more assignments than simple redundancy alone). Simple redundancy alone has no such guaranteed detection probability.

8. Related work

The Golle-Stubblebine scheme [3] described in Section 3.1 guarantees a given positive detection probability while requiring fewer resources than simple redundancy. Our distribution requires fewer resources that theirs, while providing the same effective cheat detection probability.

In recent years, a small but growing collection of papers has emerged that cover various security issues specific to these computations. Golle and Mironov [2] consider computations involving inversion of a one-way function (IOWF). They present several protection mechanisms and use game theoretic arguments to measure the efficacy of their strategies. Monrose, Wyckoff, and Rubin [6] propose instrumenting host code in order to generate lightweight execution traces that can be used to verify program execution. Most recently Sarmenta [10] and Szajda, et al. [13] present probabilistic verification mechanisms that increase the likelihood that an attempt to disrupt or cheat a computation will be detected. Neither of these papers considers the question of optimal use of resources.

More peripherally related work addresses the general problem of security in large-scale distributed computations. Wasserman and Blum [15] provide a survey of result-checking and self-correcting programs, but lack of general solutions causes applicability in our context to be limited. Many implementations of distributed computing platforms address the general issues of fault-tolerance [1, 7, 11], but assume a fault model in which errors that occur are not the result of malicious intent.

There is also a body of literature related to protecting mobile agents from malicious hosts [4, 9, 14]. However, mobile agents typically visit hosts for a short period of time, while distributed volunteer computations can require several hours of execution time — hence, time limited obfuscation approaches [4] are not applicable. Furthermore, approaches such as cryptographic traces to verify code execution [14] are not practical for large-scale distributed computations because of the storage and communication overhead required by the traces.

9. Conclusions

We have considered minimal resource task distribution schemes that guarantee a predetermined cheat detection threshold. We presented the Balanced distribution, a tunable distribution that increases collusion resistance relative to simple redundancy while decreasing the associated computational costs. It is easily deployed and requires only minor modifications to current system implementations. Moreover, the distribution is the optimal practical solution— no other scheme requires fewer total assignments without an increase in the number of precomputed tasks and/or significant decreases in non-asymptotic detection probabilities.

Acknowledgments

We would like to thank the anonymous reviewers whose comments helped us in preparing the final version of the paper. We would also like to thank Kathy Hoke for her helpful insights on the theory of linear programming.

References

[1] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wyckoff. Charlotte: Metacomputing on the web. *Future Generation Computer Systems*, 15(5-6):559–570, 1999.

[2] P. Golle and I. Mironov. Uncheatable distributed computations. In *Proceedings of the RSA Conference 2001, Cryptographers' Track*, pages 425–441, San Francisco, CA, 2001. Springer.

[3] P. Golle and S. Stubblebine. Secure distributed computing in a commercial environment. In P. Syverson, editor, *Proceedings of Financial Crypto 2001*, volume 2339 of *Lecture Notes in Computer Science*, pages 289–304. Springer-Verlag, 2001.

[4] F. Hohl. Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. In G. Vigna, editor, *Mobile Agent Security*, pages 92–113. Springer-Verlag: Heidelberg, Germany, 1998.

[5] N. Johnson, S. Kotz, and A. Kemp. *Univariate Discrete Distributions*. J. Wiley & Sons, 2nd edition, 1993.

[6] F. Monrose, P. Wyckoff, and A. Rubin. Distributed execution with remote audit. In *Proceedings of the 1999 ISOC Network and Distributed System Security Symposium*, pages 103–113, 1999.

[7] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computing over the internet—the Popcorn project. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 592–601, Amsterdam, Netherlands, May 1998.

[8] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 3 edition, 1987.

[9] T. Sander and C. F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In G. Vigna, editor, *Mobile Agent Security*, pages 44–60. Springer-Verlag: Heidelberg, Germany, 1998.

[10] L. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, March 2002.

[11] L. Sarmenta and S. Hirano. Bayanihan: Building and studying web-based volunteer computing systems using java. *Future Generation Computer Systems*, 15(5/6):675–686, 1999.

[12] The Search for Extraterrestrial Intelligence project. University of California, Berkeley. http://setiathome.berkeley.edu/.

[13] D. Szajda, B. Lawson, and J. Owen. Hardening functions for large-scale distributed computations. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 216–224, Berkeley, CA, May 2003.

[14] G. Vigna. Cryptographic Traces for Mobile Agents. In G. Vigna, editor, *Mobile Agent Security*, pages 137–153. Springer-Verlag: Heidelberg, Germany, 1998.

[15] H. Wasserman and M. Blum. Software reliability via run-time result-checking. *Journal of the ACM*, 44(6):826–849, 1997.

A. Justification of $1/\sqrt{N}$ Claim

Assume that there are N tasks, and that each task is to be distributed twice, exactly once in each of two phases. Distribution of tasks in the second phase begins after all tasks have been distributed and results collected in the first phase. Assume also that the adversary controls proportion p of participants in each phase. We show here that for reasonable (and realistic) values of p and N, the expected number of tasks of which the adversary controls both copies is well approximated by p^2N. To see this, assume that the adversary is assigned exactly k work units in each phase. Assume also that the tasks are indexed from 0 through $N-1$ and that she is assigned tasks $0, 1, \ldots, k-1$ in phase one. Now consider task assignments in the second phase. We may assume without loss of generality that the adversary is assigned the first k randomly picked tasks from the second phase. Any task she receives with index less than k is a copy of a task she received in the first phase. For large N and reasonably small k (say $k < N/2$) this is well approximated by the binomial distribution $B(k, k/N)$ with k trials and probability of success k/N in each trial. Thus the expected number of tasks of which the adversary controls both copies is k^2/N. But we assumed that the adversary controls proportion p of participants in each phase. That is, $p = k/N$, and thus the expected value is $p^2N^2/N = p^2N$. Stated another way, the adversary is expected to be able to cheat on at least one task as long as $Np^2 \geq 1$, or equivalently $p \geq 1/\sqrt{N}$.

B. Proof of Proposition 1

Proof. We provide the proof for system S. Modification for the S_k is straightforward. Consider the system derived

from S by eliminating all constraints except C_0 and C_1. Call this system S'. Because S' is derived from S by eliminating constraints, the space $F(S')$ of feasible solutions to S' contains the set of feasible solutions $F(S)$ to S. We achieve the desired result by showing that the distribution $\bar{x}$ with

$$x_1 = \frac{2N(1-\alpha)}{2-\alpha}, \qquad x_2 = \frac{\alpha N}{2-\alpha}, \qquad \text{and } x_i = 0$$

for $i > 2$ is the unique optimal solution for S'. Since $\bar{x}$ is not feasible for S, this implies that this redundancy level cannot be achieved by a feasible solution to S.

First, note that the total number of assignments required by $\bar{x}$ is

$$x_1 + 2x_2 = \frac{2N}{2-\alpha}.$$

Now, let $\bar{v}$ be any other feasible solution to S'. By constraint C_1,

$$v_1 \leq \left(\frac{1-\alpha}{\alpha}\right) \sum_{i=2}^{\infty} i v_i.$$

Equivalently (adding v_1 to both sides),

$$\frac{1}{1-\alpha} v_1 \leq \sum_{i=1}^{\infty} i v_i.$$

If $v_1 > x_1$, then

$$\sum_{i=1}^{\infty} i v_i \geq \frac{1}{1-\alpha} v_1 > \frac{1}{1-\alpha} x_1 = \frac{1}{1-\alpha} \frac{2N(1-\alpha)}{2-\alpha} = \frac{2N}{2-\alpha},$$

so $\bar{v}$ cannot be optimal. On the other hand, if $v_1 < x_1$, then

$$\sum_{i=1}^{\infty} i v_i > v_1 + 2(N - v_1) = 2N - v_1 > 2N - x_1 = \frac{2N}{2-\alpha},$$

so again $\bar{v}$ cannot be optimal. Thus an optimal solution to S' must assign exactly x_1 tasks with multiplicity 1. This in turn implies that any potential optimal solution $\bar{v}$ to S', must satisfy

$$\sum_{i=2}^{\infty} i v_i = \sum_{i=2}^{\infty} i x_i.$$

If $v_2 > x_2$, then the sum of the $\bar{v}$ is strictly greater than N. If $v_2 < x_2$, then since $x_1 + x_2 = N$ and $\sum_{i=1}^{\infty} v_i \geq N$, $\bar{v}$ must assign some tasks three times. In either case,

$$\sum_{i=1}^{\infty} i v_i > \frac{2N}{2-\alpha},$$

so $\bar{v}$ is not optimal. □

C. Proof of Theorem 1

Proof. Properties 1 and 3 follow immediately from the fact that the Balanced distribution can be expressed as a constant multiple (N) times the zero-truncated Poisson Distribution with parameter $\ln\left(\frac{1}{1-\alpha}\right)$ [5].
Property 2: From Section 2.2,

$$P_k = a_k \iff a_k = \left(\frac{1-\alpha}{\alpha}\right) \sum_{i=k+1}^{\infty} \binom{i}{k} a_i.$$

Now,

$$\left(\frac{1-\alpha}{\alpha}\right) \sum_{i=k+1}^{\infty} \binom{i}{k} a_i = N \left(\frac{1-\alpha}{\alpha}\right)^2 \sum_{i=k+1}^{\infty} \binom{i}{k} \frac{\gamma^i}{i!}$$

$$= N \left(\frac{1-\alpha}{\alpha}\right)^2 \frac{1}{k!} \sum_{i=k+1}^{\infty} \frac{\gamma^i}{(i-k)!}$$

$$= N \left(\frac{1-\alpha}{\alpha}\right)^2 \frac{\gamma^k}{k!} \sum_{i=1}^{\infty} \frac{\gamma^i}{i!} = N \left(\frac{1-\alpha}{\alpha}\right)^2 \frac{\gamma^k}{k!} (e^{\gamma} - 1)$$

$$= N \left(\frac{1-\alpha}{\alpha}\right)^2 \frac{\gamma^k}{k!} \left(\frac{\alpha}{1-\alpha}\right) = N \left(\frac{1-\alpha}{\alpha}\right) \frac{\gamma^k}{k!} = a_k$$

□

D. Proof of Proposition 2

Proof. The proof for the finite and infinite dimensional cases are similar, so we include only the proof for the infinite dimensional case here. We assume, as in the hypothesis, that p is fixed. Clearly $P_{k,p}$ is identical for all k if and only if $1 - P_{k,p}$ is identical for all k as well. By definition, $1 - P_{k,p}$ is the probability that an adversary controlling proportion p of the assignments is *not* detected when cheating on a task of which she controls k copies. So, let $\bar{x} = (x_1, x_2, \ldots)$ be a distribution that requires the minimum number of tasks from all distributions in $\mathcal{E}_\infty$. Let T be an arbitrary task, A_k be the event that the adversary has been assigned exactly k copies of this task, and let n be the multiplicity of the task (i.e., the total number of times the task has been assigned in the computation according to distribution $\bar{x}$). By the definition of conditional probability, we have

$$1 - P_{k,p} = P(n = k | A_k) = \frac{P(A_k | n = k) P(n = k)}{P(A_k)}.$$

Now, $P(n = k) = x_k / N$. Thus,

$$1 - P_{k,p} = \frac{p^k \frac{x_k}{N}}{\sum_{i=k}^{\infty} p^k (1-p)^{i-k} \binom{i}{k} \frac{x_i}{N}} = \frac{x_k}{\sum_{i=k}^{\infty} (1-p)^{i-k} \binom{i}{k} x_i}.$$

By assumption, the expression $1 - P_{k,p}$ is independent of k. Since taking the limit as $p \to 0$ preserves this property, we have that for all positive integers k

$$\begin{aligned}\lim_{p\to 0} & \frac{x_k}{\sum_{i=k}^{\infty}(1-p)^{i-k}\binom{i}{k}x_i} \\ &= \frac{x_k}{\sum_{i=k}^{\infty}\lim_{p\to 0}(1-p)^{i-k}\binom{i}{k}x_i} \\ &= \frac{x_k}{\sum_{i=k}^{\infty}\binom{i}{k}x_i} = \frac{x_k}{x_k + \sum_{i=k+1}^{\infty}\binom{i}{k}x_i}\end{aligned}$$

where the switch of the limit and the sum is justified by a straightforward application of the Lebesgue Dominated Convergence Theorem [8]. Let η be the common value of these fractions. That is

$$\frac{x_k}{x_k + \sum_{i=k+1}^{\infty}\binom{i}{k}x_i} = \eta \tag{3}$$

for all positive integers k. Clearly $0 < \eta < 1$. Also,

$$x_k = \frac{\eta}{1-\eta}\sum_{i=k+1}^{\infty}\binom{i}{k}x_i.$$

Since

$$x_k \leq \left(\frac{1-\alpha}{\alpha}\right)\sum_{i=k+1}^{\infty}\binom{i}{k}x_i$$

we must have

$$\frac{\eta}{1-\eta} \leq \frac{1-\alpha}{\alpha}.$$

The result will follow if we can show that this inequality is in fact equality. So, suppose for contradiction that

$$\frac{\eta}{1-\eta} < \frac{1-\alpha}{\alpha}.$$

Then $1 - \eta > \alpha$. Now, the expression on the left hand side of (3) is the asymptotic probability that the adversary is *not* caught while trying to cheat on a task of which she possesses k copies. Thus the asymptotic probability that the adversary *is* caught trying to cheat on a task of which she possesses k copies is $1 - \eta$. But $1 - \eta > \alpha$ which implies that the minimal distribution of tasks satisfying all constraints at detection threshold α actually satisfies the constraints for the higher threshold $1 - \eta$. This is a contradiction, since (all other factors being equal) the total number of assignments required is a strictly increasing function of α. □

E. Proof of Proposition 3

Proof. We copy the technique used in [3]. Choose a positive integer k and let A_k be the event that the adversary has been assigned a given task k times. Let n be the multiplicity of the task. Then

$$1 - P_{k,p} = P(n = k|A_k) = \frac{P(A_k|n=k)P(n=k)}{P(A_k)}.$$

If we let $\gamma = \ln\left(\frac{1}{1-\alpha}\right)$, then $P(A_k|n=k) = p^k$,

$$P(A_k) = \sum_{i=k}^{\infty} p^k(1-p)^{i-k}\binom{i}{k}P(n=i),$$

and

$$P(n=i) = \left(\frac{1-\alpha}{\alpha}\right)\frac{\gamma^i}{i!}.$$

So

$$\begin{aligned}1 - P_{k,p} &= \frac{p^k\left(\frac{1-\alpha}{\alpha}\right)\frac{\gamma^k}{k!}}{\sum_{i=k}^{\infty}p^k(1-p)^{i-k}\binom{i}{k}\left(\frac{1-\alpha}{\alpha}\right)\frac{\gamma^i}{i!}} \\ &= \frac{\frac{\gamma^k}{k!}}{\sum_{i=k}^{\infty}(1-p)^{i-k}\binom{i}{k}\frac{\gamma^i}{i!}} \\ &= \frac{\frac{\gamma^k}{k!}}{\sum_{i=k}^{\infty}(1-p)^{i-k}\frac{i!}{k!(i-k)!}\frac{\gamma^i}{i!}} \\ &= \frac{(\gamma(1-p))^k}{\sum_{i=k}^{\infty}(1-p)^i\frac{\gamma^i}{(i-k)!}} \\ &= \frac{(\gamma(1-p))^k}{\sum_{i=0}^{\infty}\frac{(\gamma(1-p))^{i+k}}{i!}} \\ &= \frac{(\gamma(1-p))^k}{(\gamma(1-p))^k\sum_{i=0}^{\infty}\frac{(\gamma(1-p))^i}{i!}} \\ &= \frac{1}{e^{(1-p)\gamma}} = (1-\alpha)^{1-p}\end{aligned}$$

The result follows. □