

REPORT DOCUMENTATION PAGE*Form Approved*
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**1. REPORT DATE (DD-MM-YYYY)**

AUGUST 2011

2. REPORT TYPE

Conference Paper-Post Print

3. DATES COVERED (From - To)

May 2009 – November 2010

4. TITLE AND SUBTITLEANALYSIS OF BINARY VOTING ALGORITHMS FOR USE IN
FAULT-TOLERANT AND SECURE COMPUTING**5a. CONTRACT NUMBER**

In House

5b. GRANT NUMBER

N/A

5c. PROGRAM ELEMENT NUMBER

62702F

6. AUTHOR(S)Kevin Kwiat, Alan Taylor, William Zwicker, Daniel Hill, Sean Wetzonis,
Shangping Ren**5d. PROJECT NUMBER**

23G4

5e. TASK NUMBER

IH

5f. WORK UNIT NUMBER

01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)AFRL/Information Directorate Union College
Rome Research Site/RIGD Mathematics Department
525 Brooks Road Schenectady NY 13208
Rome NY 13441**8. PERFORMING ORGANIZATION
REPORT NUMBER**

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)Air Force Research Laboratory/Information Directorate
Rome Research Site
26 Electronic Parkway
Rome NY 13441**10. SPONSOR/MONITOR'S ACRONYM(S)**

AFRL/RI/RRS

**11. SPONSORING/MONITORING
AGENCY REPORT NUMBER**

AFRL-RI-RS-TP-2011-3

12. DISTRIBUTION AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA #: 88ABW-2010-1912.

Date Cleared: April 7, 2010.

13. SUPPLEMENTARY NOTES

© 2010 IEEE. This article appeared in the Proceedings of the 2010 IEEE International Conference on Computer Engineering and Systems (ICCES). This work is copyrighted. One or more of the authors is a U.S. Government employee working within the scope of their Government job; therefore, the U.S. Government is joint owner of the work and has the right to copy, distribute, and use the work. All other rights are reserved by the copyright owner.

14. ABSTRACT

This paper examines three binary voting algorithms used with computer replication for fault tolerance and separately observes the resultant reliability and security. The paper aims to offer insights to answer the question: Can a voting algorithm provide a system with both security and reliability? The papers show that while random dictator (i.e., randomly choosing one of the replicas) provides good security and majority rule yields good fault tolerance, neither is effective in both. The random troika (a subset of 3 replicas) as an effective combination of fault-tolerant and secure computing.

15. SUBJECT TERMS

Fault Tolerance, Reliability, Game Theory, Voting

16. SECURITY CLASSIFICATION OF:**a. REPORT**

U

b. ABSTRACT

U

c. THIS PAGE

U

**17. LIMITATION OF
ABSTRACT**

UU

**18. NUMBER
OF PAGES**

6

19a. NAME OF RESPONSIBLE PERSON

Kevin A. Kwiat

19b. TELEPHONE NUMBER (Include area code)

N/A

Analysis of Binary Voting Algorithms for use in Fault-Tolerant and Secure Computing

Kevin Kwiat*, Alan Taylor[†], William Zwicker[†], Daniel Hill[‡], Sean Wetzonis[§], Shangping Ren[¶]

*Cyber Science Branch

Air Force Research Laboratory, Rome, New York

[†]Mathematics Department, Union College, Schenectady, New York

[‡]Mathematics Department, Bethel College, South Bend, Indiana

[§]Dept of Electrical and Computer Engineering, University of Massachusetts at Lowell, Lowell, Massachusetts

[¶]Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois

Abstract—We examine three binary voting algorithms used with computer replication for fault tolerance and separately observe the resultant reliability and security. We offer insights to answer the question: Can a voting algorithm provide a system with both security and reliability? We show that while random dictator (i.e., randomly choosing one of the replicas) provides good security and majority rule yields good fault tolerance neither is effective in both. We present the random troika (a subset of 3 replicas) as an effective combination of fault-tolerant and secure computing.

I. INTRODUCTION

Computers are often replicated to mask out failures, with some form of voting employed to reach agreement on the data fielded by the various replicas [2]. Given a set of data each computer must independently come to its own conclusion, compare its conclusion to the other computers conclusions, and communicate the decisions of the entire set so that the decision representing the set is posted as the result.

In this paper, we work with only symmetric binary voting systems. A binary voting system is a system that produces either an incorrect solution or a correct solution. Typically, in useful binary voting systems, there is a given probability p greater than 0.5 that any individual decision is correct.

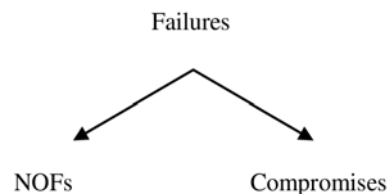
A coin flipping game with $p = 0.5$ models an unsure binary voting system because the outcome has no propensity towards either result.

In critical systems, voting algorithms are used because the correctness of a solution cannot be trusted to a single computer.

A replicated computer will always choose the outcome it determines is the most correct, but this choice may be based on data that is incomplete or vulnerable.

Therefore, it is a possibility that this computer may still vote incorrectly even if it is fault-free. For this reason, we use a value p as the probability of a correct response by a non-failed machine, i.e., the reliability of a machine. In addition to a computer choosing incorrectly due to insufficient data, a computer may also vote incorrectly due to some kind of fault.

A reliable system is fault-tolerant or, in other words, is capable of sustaining correct operation in the presence of *naturally-occurring faults* (NOFs). We restrict the definition of security to attack tolerance: the capability of continuing correct operation after a non-random, directed fault occurs (the system is attacked). We will call these directed faults *compromises*. Both NOFs and compromises are examples of *failures*. A computer experiencing either a NOF or a compromise will be called a *failed machine*.



Our motivation for combining security and fault tolerance comes from [3]. The aim of this paper is to investigate the simultaneous treatment of compromises and NOFs and evaluate the effectiveness of different voting algorithms on the survivability of a system when both types of failures may occur.

II. BINARY VOTING

Many voting algorithms require a central tallyer (often times called the *voter*) that collects the individual results from the replicated computers and produces the decision. However, distributed voting algorithms exist that do not require a central tallyer, such as in [1], but treats each replica as an individual voter that is capable of casting the final decision. Therefore, in our paper, we treat the replicated computers as having the capability to perform distributed voting that does not require a central tallyer.

The type of voting we are concerned with is binary voting. A simple binary voting system can be modeled by stating the required number of votes to cause a positive outcome. For example, the notation $[N; V_1, V_2, V_3, \dots]$ defines a voting pool where N is the number of votes required to create a positive outcome, and $V(1, 2, 3, \dots)$ is the weight of the voter in that spot. The system $[3; 1, 1, 1, 1, 1]$ indicates there are five

*Approved for Public Release; Distribution Unlimited: 88ABW-2010-1912 dated 07 Apr 2010

voters each with a weight of one. If any three of them agree, then the decision is made.

The example system above is anonymous because each voter has the same expected contribution as any other voter. The following systems assume anonymity:

Majority Rule (MR): [3; 1, 1, 1, 1, 1]

Majority rule requires one more vote than half the number of voters in order to make a decision. All voters must have the same weight.

MR exhibits high reliability, but low security. If the number of failed computers (a computer that has experienced either a NOF or compromise) is exactly zero, or is low compared to the total number of computers, MR will choose correctly with a high degree of certainty. The certainty increases as the number of computers increases. According to the Condorcet Jury Theorem [4], as the number of voting computers approaches infinity, the odds that majority rule chooses correctly approaches 1.

Random Dictator (RD): [1; 0, 0, 0, 0, ... 1]

The RD method randomly selects one computer to make the decision for the group. The number of voting computers is always one.

RD exhibits a high degree of security but relatively low reliability. If the number of failed computers approaches or is equal to zero and the system is small, then RD performs similarly to MR. As the number of voting machines approaches infinity the probability that random dictator chooses correctly approaches p .

Random Troika (RT): [2; 1, 1, 1, 0, 0]

Random troika, named from the Russian word for a trio of horses, randomly selects three machines to vote by majority. The majority of the troika is then committed as the decision of the system. Figure 1 shows two clusters of 9 computers each where one cluster operates under RD and the other operates under RT.

Random troika essentially reduces the size of the voting pool with the goal of increasing the average value p within the smaller pool.

In any system it is hoped that the value of p will be high and that computers will be without failures; however, this is not always the case. The following examples calculate the reliability of a cluster of five computers employing each of the above presented voting algorithms given 0, 1, and 2 NOFs. Each example shows the outcome given all values of p between [0,1].

A. MR vs. RD vs. RT assuming 0 failures

The reliability function calculates the probability that a cluster decides correctly. For MR, in the ideal case of zero failures, the reliability function is calculated by taking the probability that three computers vote correctly (p^3) multiplied by the number of combinations of the three computers $\binom{5}{3}$

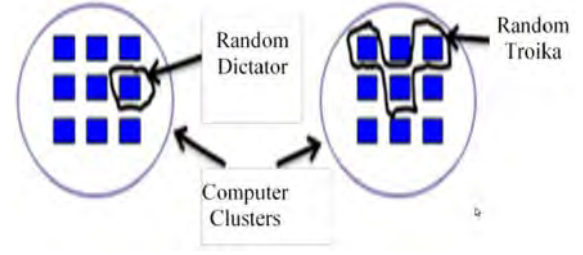


Fig. 1. Random Dictator (RD) vs. Random Troika (RT)

and multiplied by the probability of two incorrectly voting computers $[(1-p)^2]$:

$$10p^5 - 20p^4 + 10p^3 \quad (1)$$

Then we must add the probability that any four computers vote correctly (p^4), multiplied by the combinations of four computers $\binom{5}{4}$, multiplied by the probability of one fault-free computer voting incorrectly $(1-p)$:

$$5p^5 - 15p^4 + 10p^3 \quad (2)$$

Lastly the probability that all five vote correctly (p^5) must be added to (2), which simplifies to:

$$6p^5 - 15p^4 + 10p^3 \quad (3)$$

Therefore, equation 3 is the reliability function of a system with a MR algorithm.

The reliability function for RD, however, is the probability that any one computer selects correctly, or simply:

$$p \quad (4)$$

The reliability function for RT is:

$$-2p^3 + 3p^2 \quad (5)$$

For a system without failures, MR will decide correctly more often than RD for values of $p > 0.5$. This indicates that MR is superior in cases of five or more voters for values of $p > 0.5$. The probability of deciding correctly in a RT system falls between that of MR and RD for all p .

Thus the reliability functions for systems with no failures are:

- MR: $6p^5 - 15p^4 + 10p^3$
- RD: p
- RT: $-2p^3 + 3p^2$

The graph in Figure 2 shows how their reliabilities compare over all values of p .

B. MR vs. RD vs. RT assuming 1 NOF

For cases where one computer is known to have a failure, the MR reliability function is calculated in the same method as in the last example and results in:

$$-3p^4 + 4p^3 \quad (6)$$

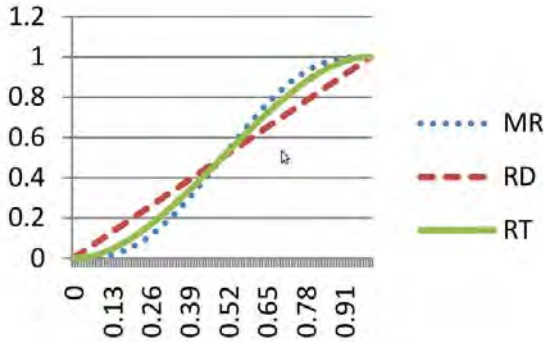


Fig. 2. MR vs. RD vs. RT in zero failures case (x-axis: values of p , y-axis: reliability)

RD has a reliability function of:

$$\frac{4p}{5} \quad (7)$$

which is calculated by taking the probability that a non-failed computer is chosen ($\frac{4}{5}$) multiplied by the probability that it votes correctly (p).

RT reliability function is:

$$-\frac{4p^3}{5} + \frac{9p^2}{5} \quad (8)$$

RT, while sub-optimal for all values of p , is greater than MR for values less than approximately 0.6 and greater than RD for values greater than 0.6. Integrated over $p = (0, 1]$ shows the average reliability. It is surprising to note that RT has the greatest average value which makes it useful for cases where p is either random or unknown. The graph for one NOF is very similar to that of zero failures.

Figure 3 shows a comparison of the reliability functions of the voting algorithms.

- 1) If $0.5 < p < 0.600$, then $RD < RT < MR$
- 2) If $0.600 < p < 0.6051$, then $RD > MR > RT$
- 3) If $0.6051 < p < 0.6096$, then $MR > RD > RT$
- 4) If $0.6096 < p < 1$, then $MR > RT > RD$

(Note: The values 0.600, 0.6051, and 0.6096 are accurate to four decimal places and are the intersections of the three equations.)

If the precise value of p is known, then RT is never optimal; it would then seem logical to dismiss RT. However, if the value of p is unknown, then the natural approach is to choose the algorithm whose reliability function has the greatest average value on the interval $[0, 1]$. This is calculated by taking the integral of the functions over the interval $[0, 1]$. RT then becomes the logical choice; however, in more realistic cases where $p > 0.5$, MR is the most logical choice. Nevertheless, we will see that RT does bare an unexpected significance.

C. RD vs. MR vs. RT assuming two failures

Recall that as the number of failed computers increased the probability of a correct decision in an MR system decreased.

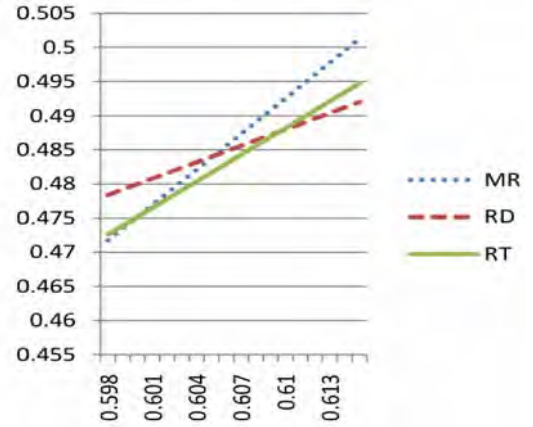


Fig. 3. MR vs. RD vs. RT from $p = [0.595, 0.613]$ in one failure case (x-axis: values of p , y-axis: reliability)

MR was optimal for values of $p > 0.61$ with zero compromises and for $p > 0.77$ with one compromise. The reliability functions of the three methods assuming two failures are:

- MR: p^3
- RD: $\frac{3p}{5}$
- RT: $-\frac{p^3}{5} + \frac{9p^2}{10}$

Performing a similar analysis as in the previous cases we obtain the following:

- 1) If $0 < p < 0.75$, then $RD > RT > MR$
- 2) If $0.75 < p < 0.6547$, then $RD > MR > RT$
- 3) If $0.6547 < p < 0.8139$, then $MR > RD > RT$
- 4) If $0.8139 < p < 1$, then $MR > RT > RD$

The same conclusion can be made as in the one failure case: RT is not an optimal strategy. We will see, however, that to abandon RT after only examining its effect on reliability would be premature.

III. STRAGIC CONCLUSION

The above examples evaluate the *reliability* provided by three distinct voting algorithms when a system experiences 0, 1, and 2 NOFs; however, these examples fail to consider the *security* of the algorithms.

We begin by examining a smaller scenario of two clusters of three computers each and an attacker versus a defender. Since participants in a competition are adversaries, their strategies are opposed to one another; a positive outcome for one is the exact opposite for the other. We assume that players in competition will rationally take courses of action that maximize the likelihood of their desired outcome.

A failed cluster is one that will produce an incorrect output after voting, and such a cluster constitutes a loss for the defender and a win for the attacker, making the game zero-sum. In this scenario p is assumed to be one. The defender will choose either RD for both clusters or MR for both clusters. The attacker may compromise any two computers, resulting in the event matrix of Figure 4.

Figure 4 displays all possible outcomes of an attack on two clusters with two scenarios of attack strategies and two

possible algorithms for defense. The 2, 0 column represents the scenario of two compromises in the first cluster and zero in the second, and the 1, 1 column represents the scenario of one compromise in each cluster. The values in the graph represent the expected number of clusters to survive given the defense choice and the attack strategy.

		Attack Strategy	
		{2,0}	{1,1}
Defense	MR	1	2
	RD	4/3	4/3

Fig. 4. Attacker vs. Defender Event Matrix

If the attacker is rational, he recognizes that attack 2, 0 is the dominant strategy because attack 1, 1 is at least as good as the alternative strategy for both defense choices. If both the attacker and defender make such a matrix they are both able to determine their best strategy. The attacker will not choose 1, 1 because the defender will then choose MR and the attacker will not be able to compromise any clusters. The defender will choose RD since 4/3 is greater than 1. The defender recognizes that his losses are minimized when RD is selected.

We now expand the scenario to a defender who must protect nine clusters of nine computers each ($9 \times 9 = 81$ computers) and the attacker may compromise 25 computers. As with the previous scenario, a failed cluster is one that will produce an incorrect output after voting, and such a cluster constitutes a loss for the defender and a win for the attacker. p is still assumed to be one.

If the attacker spreads his 25 attacks roughly equally among the clusters, then each cluster loses 2 or 3 computers out of 9. This is less than a majority, so MR will make a correct decision in all 9 clusters. If the attacker chooses 5 clusters and compromises 5 computers from each cluster, then MR will decide incorrectly in 5 out of the 9 clusters (i.e., a majority of clusters). A rational attacker will choose the latter method of compromising a bare majority of computers in as many cluster as possible given that the defender chooses MR. The attacker in the above example is exploiting a weakness of MR referred to as the “multiplier effect”.

In the previous scenario of 9 clusters we witnessed the multiplier effect and how the security of MR decreases. What happens if the defender employs RD? When the defender chooses RD then on average 259 of the 9 clusters will vote incorrectly, leaving 569 voting correctly. So the average performance of RD is much better than that of MR. The question arises, if the defender uses RD, will the attacker change his attack strategy again? It turns out that no attack strategy is more effective against RD than any other, so RD holds the proportion of failed clusters down to the proportion of failed computers, denying the enemy the advantage he has when the defender uses MR. Thus the best strategy for the attacker is to compromise 5 computers in 5 clusters and the defender should always choose RD to deny the attacker the multiplier effect.

We saw earlier that MR improves fault-tolerance. Is it possible that our assumption that $p = 1$ skews our results in favor of RD over MR? The Condorcet Jury Theorem suggests that the effectiveness of RD will be degraded by a greater amount than MR as p decreases because the fault tolerance of MR is better than that of RD. Thus, RD is better at security while MR is better at reliability.

We now reintroduce RT and look at its effect on the security of these 9 clusters. RT combines a degree of fault tolerance with some of the shell-game effect of RD. If the attacker compromises a bare majority of computers in as many clusters as possible (which is his best strategy for either RD or MR), then RT would outperform MR in the attacked clusters. MR consistently fails in these clusters while RT still has a possibility of surviving and providing the correct decision. RD, however, will do better than RT since RD provides a better probability (i.e., 4/9) of choosing correctly; yet, in the remaining unattacked clusters (i.e., clusters in which there are no attempted compromises) RT will outperform RD, but still be outperformed by MR, as shown in Figure 2 for the more likely values of $p > 0.6$. We compare the effectiveness of the algorithms with these inequalities:

Attacked clusters: $RD > RT > MR$

Unattacked clusters: $MR > RT > RD$

It is important to note that the defender cannot anticipate, a priori, which of the clusters will be attacked and which will not. Therefore, the defender cannot choose RD for the attacked clusters and MR for those that are not attacked; instead by choosing RT for all the clusters, the defender blends the security of a moving target with MRs power-to-overwhelm faulty computers. A brief analysis substantiates this claim. In each of the attacked clusters, RD produces a correct output with probability $4/9 = 0.444$; whereas RTs probability of producing a correct decision (assuming $p = 1$) for a 9 computer cluster is: the probability of selecting 3 uncompromised computers; or selecting 2 uncompromised computers and 1 compromised computer. This equates to:

$$\frac{\binom{4}{3}}{\binom{9}{3}} + \frac{\binom{4}{2}\binom{5}{1}}{\binom{9}{3}} = 0.404$$

Now, let us consider p as the individual computers reliability (i.e., the computers ability to withstand a NOF) and examine the combined security and reliability of a simplified 2 cluster system. Using RD for both clusters, the attacked cluster produces a correct decision with probability 0.444 and the unattacked cluster produces a correct decision with probability p . The joint probability that both clusters produce the correct decision is $0.444p$. Similarly, for RT, the joint probability of a correct decision being produced by the two clusters is $0.404(p^3 + 3p^2(1 - p))$. From these expressions we find, by setting the difference of the two joint probability expressions equal to 0 and finding the roots of the resultant equation, that for values of p , $0.636 < p < 0.863$, the RT-based system is superior to the RD-based system.

This simple two-cluster example demonstrates the ability of RT to effectively combine security and fault tolerance over a significant range of p values. In scenarios involving more clusters and where the individual clusters are composed of more computers, RT can be a prudent compromise between the game theoretic measures to counter an attackers optimum strategy and the engineering principles used to create digital systems that continue operating correctly even in the presence of naturally induced faults.

IV. CONCLUSION

Some voting algorithms fare better in high assurance situations than others. Majority rule has high fault tolerance but relatively low security while random dictator exhibits the opposite. We found that the random troika strikes a medium: it combines the fault tolerance attributes of majority rule with a degree of random dictators security. Considering its versatility, random troika should be among the top choices of voting algorithms for high-assurance computing applications.

In the context of voting systems, conjoining reliability (vis-à-vis fault tolerance) and security can be done reliably and securely but only with an understanding of when and how the different voting systems should be applied.

REFERENCES

- [1] B. Hardekopf and K. Kwiat. Distributed voting for security and fault tolerance. Technical Report AFRL-IF-RS-TR-2001-53, Air Force Research Laboratory, may 2001.
- [2] P. Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall, 1994.
- [3] K. Kwiat. Can reliability and security be joined reliably and securely? pages 72 –73, 2001.
- [4] A. Taylor and W. Zwicker. *Simple Games: Desirability Relations, Trading, Pseudoweighting*. Princeton University Press, New Jersey, 1999.