1997, pp. 109–115

A Different Look at Secure Distributed Computation

Paul F. Syverson
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375
(syverson@itd.nrl.navy.mil)

Abstract

We discuss various aspects of secure distributed computation and look at weakening both the goals of such computation and the assumed capabilities of adversaries. We present a new protocol for a conditional form of probabilistic coordination and present a model of secure distributed computation in which friendly and hostile nodes are represented in competing interwoven networks of nodes. It is suggested that reasoning about goals, risks, tradeoffs, etc. for this model be done in a game-theoretic framework.

1. Introduction

Models for distributed computation typically make use of a fault model for messages and/or the nodes sending and receiving messages. Some typical kinds of faults are: crash failure, in which a faulty node sends no further messages after failure (this is sometimes called fail-stop), omission faults, in which messages from a sender are not received by the intended recipient (this can be modelled as a fault at the sender, at the receiver, or in the connection between them), and byzantine faults [13, 10]. In a byzantine failure a faulty node might do virtually anything of which it is computationally capable, including altering or substituting messages, capturing or misdirecting messages that it was to forward, etc.

Modelling secure computation has generally been based on some form of worst-case assumption about the computing environment. Thus, byzantine failure is the natural failure model to assume when modelling secure distributed computation, e.g., [14, 15, 3]. Some researchers have looked at hybrid fault models, where different types of faults may occur together [4], and others have taken a broader look at relaxing the worst-case assumption approach to all areas of secure computing [9, 12]. Still, the worst-case view dominates the secure computing literature in general and the secure distributed computing literature in particular.

In section 2 we look at byzantine failure with respect to weaker than usual agreement goals, and we introduce the notion of viewing secure distributed computing as competing networks. In section 3 we look at topological considerations motivated by previous discussion. In section 4 we look at reasoning about competing networks in a game theoretic framework. In section 5 we present concluding remarks.

2. Weakening Agreement Goals and Assumptions

Byzantine agreement protocols, (protocols for acheiving agreement in the presence of byzantine failures) make two broad types of assumptions: the percentage of nodes that are byzantine faulty is less than some maximum, and the connectivity of the set of all nodes exceeds some minimum. (The connectivity of a graph is the minimum number of nodes that must be removed to partition the network.) Without any further assumptions, if there are k byzantine faulty nodes in the network, then there must be $3k + 1$ or more total nodes in the network, and the network must have connectivity of at least $2k + 1$.

We have not yet said what we mean by 'agreement'. Without going into detail, agreement includes at least that all nonfaulty nodes arrive at the same value for some chosen variable (agreement) and that this value was initially chosen by a nonfaulty node (validity). Our goal in stating conditions and goals for byzantine agreement is simply for contrast. So, we will not describe this rich area in any further detail. We here instead look at weaker kinds of agreement.

Much has been written about the problem of authenticated key distribution. Almost all analyses of

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1997		**00-00-1997 to 00-00-1997**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Different Look at Secure Distributed Computation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Center for High Assurance Computer Systems,4555 Overlook Avenue, SW,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**7**	

this problem have assumed that a hostile enemy has complete control of the network: all messages are sent to this enemy (regardless of the intended recipient) and all messages are received from this enemy. This can be viewed as a special case of byzantine failure. We can view this model as a network with a star topology in which there is one byzantine node at the center of the network. With a connectivity of 1, this is not just byzantine failure, but byzantine failure at the worst possible place. It is somewhat surprising that any kind of agreement can be reached in this environment. Indeed, Anderson and Needham have described solving such problems as "programming satan's computer" [1]. However, authenticated key distribution protocols that work do seem to solve agreement as stated above. The principals arrive at the same key to use for communication, and the key was chosen by one of the principals or a key server. (We are describing key distribution here, not Diffie-Hellman type key agreement.) How is this possible? Such protocols are not designed to guarantee anything as strong as synchronized agreement. Rather they guarantee that if principals assign any value at all for the session key, then they will agree on that value. Byzantine agreement would require that they do all assign a value. For security purposes, it is possible that we require weaker forms of agreement even in the face of byzantine failures. We will now also explore whether it is necessary to even assume a byzantine failure model for secure computing.

2.1 Hostile Failure Can Be Weaker than Byzantine Failure

This section takes a step back from worst-case reasoning, not just on connectivity, but also on other features. Byzantine failure originally arose in looking at dependable computing. It was seen to be relevant to secure computing because of its worst-case capability: since nodes can do anything, they can do the worst possible thing that a hostile agent might want. But, maybe the hostile agents are not themselves able to do the worst possible things to the network, even if they would so desire.

In some cases it is possible to design protocols that guarantee agreement with an arbitrarily high probability (< 1), even given the topology we described for key distribution [7]. However, this requires the assumption of a fixed bound on the probability of message failure. If we can design our systems so that this assumption can be sustained even in the face of hostile attack, then the result might be applicable to security. (This might build on the idea of applying dependability concepts to security [12].) But, it does not fit within current worst-case assumptions (byzantine failure). Though perhaps ultimately applicable to closed networks, over open networks such assumptions may be difficult, if not impossible, to sustain. Need we therefore assume byzantine failure? In this section we will explore two ways of weakening our assumptions of what constitutes hostile failure.

2.2 Example 1: The Coordinated Attack Problem

A version of the following problem was first described by Gray in [5] where it was called the "generals paradox". Since then it has generally been called the coordinated attack problem. (The following version is quoted from [6], pp. 555–6.)

> Two divisions of an army are camped on two hilltops overlooking a common valley. In the valley awaits the enemy. It is clear that if both divisions attack the enemy simultaneously, they will win the battle; whereas if only one division attacks, it will be defeated. The divisions do not initially have plans for launching an attack on the enemy, and the commanding general of the first division wishes to coordinate a simultaneous attack (at some time the next day). Neither general will decide to attack unless he is sure that the other will attack with him. The generals can only communicate by means of a messenger. Normally, it takes the messenger one hour to get from one encampment to the other. However, it is possible that he will get lost in the dark or, worse yet, be captured by the enemy. Fortunately, on this particular night, everything goes smoothly. How long will it take them to coordinate an attack?

The standard claim is that, even if everything does go smoothly, no agreement can ever be reached and thus neither general can ever decide to attack. As Halpern and Moses point out this is a virtual folk theorem of operating systems theory. Suppose that general A sends a message saying "Let's attack at dawn." to general B. This is enough for B to know that A wants to attack at dawn. But, B also knows that A can't know that he knows this because the messenger might not have arrived. So, he sends back his own messenger telling A of his receipt of the message and his agreement. To indicate that everything is confirmed A acknowledges receipt of this message by sending a response to B. It might seem that the attack is now coordinated because both A and B know that they each

want to attack at dawn. And, each of them knows that they both know this. The problem is that A cannot know that his last message to B got through. So B must send an acknowledgement. But how does B know that this message arrived? A must send another acknowledgement .... An easy induction argument shows that no number of messages is sufficient to coordinate the attack.

2.2.1 Probabilistic Coordinated Attack

Halpern and Tuttle show how to achieve a probabilistic coordinated attack in the event that we can assign a fixed bound on the probability of message failure [7]. Their protocol is quite simple. Assume, for example, that the probability of message failure is less than $1/2$. Suppose A sends k messages to B, each announcing her intent to attack at dawn. A then attacks at dawn. And, B attacks at dawn if he gets A's message. The probability that the attack will be coordinated is at least $1 - 1/2^k$. By choosing suffiiciently large k given that we know the bound on message failure, the probability of coordination can be made arbitrarily close to 1.

2.2.2 Probabilistic Agreement in a Hostile Environment

In a hostile environment, it may not be reasonable to assume that we can give a fixed bound on the probability of message failure. This would seem to rule out any form of agreement. However, while we may not be able to assign even a probability to adversary behavior, we can back off of the virtual omnipotence assigned to adversaries in the case of byzantine failure. Specifically, we might assume that there is some (nonzero) chance that each message will go through even if we cannot put a lower bound on what that chance is. In this case, a nontrivial form of coordination is still possible. Specifically, the generals can guarantee that if either attacks, then the probability that the other attacks is arbitrarily close to 1.

The protocol is as follows: Instead of A sending several messages at once to B, A sends a single message to B announcing her attack time and a threshold value k. (We assume messages are confidential, correct, i.e., integrity protected, and not replayable without detection. We will not discuss here the mechanisms to achieve those assumptions.) B then responds, and A responds to the response, etc. Each message is sent if and only if the previous message is received, up to some number of messages n. And, A and B are to keep sending messages until either the attack time is past or some number of messages $n + 1$ has been sent. At this point A and B are to evaluate the number of messages received. (Recall that the adversary may prevent a message, hence all future messages, from arriving.) If the number of messages a general receives is k or greater, then the general attacks. Otherwise, he stays. If $1 \leq k \leq n$, then the probability that a given k is the threshold value is $1/n$. So the probability that the generals were discoordinated (one attacked and the other did not) is

$$(\sum_{k=1}^{n} \Pr(\text{ Adversary chooses } k \;\&\; k \text{ is threshold})) \leq 1/n$$

Thus, the probability that if one general attacks, the other will too is at least $1 - 1/n$. Note that for efficiency A and B might have a timeout mechanism that would allow either to terminate the protocol if no message is received more than some interval after the last previous one was sent. But, the basic protocol

Aside from providing a new protocol with a new capability, the above shows that hostile failure need not imply an all powerful adversary (as in byzantine failure). Though we cannot assign any probability to the attack succeeding, in our hostile failure model we can say that there is some possibility that the attack will proceed and the probability of discoordination can be made arbitrarily small. If, as in byzantine failure, we assume that the adversary can definitely prevent the attack, the only motivation for the protocol would be if the adversary's desire for the chance to cause discoordination outweighs his desire to have no attack at all. In other words, if the adversary is not capable of blocking all messages with certainty (or even if he is tempted by the prospect of causing discoordination to allow the possibility of coordination), then it is more realistic to assume sent messages will have nonzero probability of arriving than that the adversary has complete control over sent messages. Our next example also shows that a hostile adversary need not be so strong as to be able to cause byzantine failure, but has a different network topology and different goals.

2.3 Example 2: Competing Networks

The basic byzantine agreement model we have described represents distributed computation as taking place on a homogeneously composed network of nodes with a common purpose and in which some unknown number of nodes has turned buggy or malicious. The question then is to see what is necessary for the remaining nodes to accomplish their goal. The model we now suggest represents distributed computing as two or more interwoven networks of competing nodes. In

the simplest (two network) case each network has its own goals, which typically run contrary to the goals of the competing network.

This is a generalization of what we sketched in connection with authenticated key distribution and in more detail when discussing coordinated attacks. In those cases, all enemy nodes are assumed to be in direct communication, perfectly synchronized, etc. Here we can consider that they have agreement problems of their own to solve, problems that may be the result of actions taken by the friendly nodes.

Consider a network in a ring topology. The network is partitioned into two competing subnetworks such that between any two nodes of the 'good' network is a node of the 'evil' network, and vice versa. Suppose that the evil nodes would like to produce some insecure system state and that this can only be achieved if they act in concert. (For example, perhaps they want to change the access control on some object or synchronize on some clock for use in covert communication.) Assume that there is more than one combination that will yield the insecure state and that the agents know this; they simply need to coordinate on one combination.

Let us suppose that the evil network is composed of n nodes and that there is exactly one relevant choice for each node to make. They can each set some value to 1 or 0. If they all choose 1 or all choose 0, they will coordinate. Otherwise, they will not. If the evil subnetwork is characterized by byzantine failure, then the network as a whole is insecure. Since byzantine nodes are capable of any action, they certainly can all produce a 1 or all produce a 0. But, since there is a good node between every two evil ones, choosing together is problematic. Even if messages from one evil node to another are encrypted, the good nodes can simply intercept them. There is no way for the evil nodes to directly communicate so as to decide on 0 or 1. And, simply guessing, there is no better than $1/2^{n-1}$ chance of coordinating to produce the insecure state. Assuming that this is within acceptable risk limits, the system may be considered secure.

Several points are illustrated by this example. First, the model is a mixture of worst-case and less than worst-case assumptions. That there is an evil node between any two good nodes is as bad as can be from the perspective of good nodes hoping to agree in any way. On the other hand, that there is a good node between any two evil ones assumes that connectivity for the evil nodes is no better than for the good nodes. Second, the example introduces probability into our security considerations. This is not new in itself. But, it points in the direction of calulated risks and tradeoffs rather than absolute security. In the next section we will look at a mathematical framework in which to couch those risks for competing networks. Third, the example shifts the perspective from secure computing in the face of hostile attack, to hostile computing in the face of system limitations, whether or not these be countermeasures to hostile computing. Ideally, we would like to take the more realistic view encompassed in a combination of both perspectives. In the example, the evil nodes may be unable to coordinate, but the symmetry of the network means that the same is true of the good nodes. Further work might focus on the different goals of secure computing and hostile computing. This might lead to recommendations for network topology and other design issues to enhance secure computing goals. We now turn to the question of network topology.

3. Topology Considerations

Even if the network of the last example were arranged as described, how could the nodes know this? One answer is that they might just know in a particular case; the network might actually be constructed this way. (In this case, good and evil nodes are probably more accurately called 'trusted' and 'untrusted' respectively.) The example places one good node between each evil node in order to make a point about the nature of hostile failure. But, viewing the network in this way is useful for security even if the nodes are not arranged in this way.

For security purposes, any untrusted nodes directly connected by an untrusted path may effectively be viewed as a single adversary. (Recall this is the basis for sending all messages through a single adversary in the model of key distribution protocols.) Similarly, any trusted nodes connected by a trusted path may be viewed as capable of complete synchronization. What about evil nodes talking on good paths or vice versa? We can consider the paths to simply be other network components. Thus, in a graph representing the network they become nodes as well. In other words, for the purposes of a graph representing network security, anything that can be labelled good or evil is a node. Edges in the network security graph are not under anyone's control, and a message sent to an adjacent node in the security graph is always received immediately and without alteration.

An immediate consequence of this view of security is that any network security graph is bipartite (two-colorable). This simplification can give us a quick picture of some of the capabilities of both the trusted and untrusted components in the network. If we want to

control the communication of untrusted components then they cannot be at the same node of the security graph. Similarly, the coordination and communication of trusted components not at the same node of the security graph may be subject to hostile interference. These are very simple observations. More subtle results from the theory of bipartite graphs may inform our understanding of adversary capability as well as our decisions about security resources, both in the minimum total trusted components required for a given goal and in the placement of those components. We explore another advantage of this simplified view in the next section.

4. Secure Computing as Game Playing

In talking about computer security we often refer to an adversary, enemy, intruder, hostile agent, etc. However, while system design often involves a threat model, the approach is to design the system to accomplish or prevent certain goals no matter what the adversary does. Part of the proposal here is to take the idea of adversary literally and to model secure computation as a game between adversaries.

A first advantage of this approach is to allow a shift from the typical worst-case reasoning to something more flexible. In a two-person game, the players have a fixed set of strategies, and each pair of strategies gives rise to an outcome. The players each have a payoff attached to each outcome. (Actually it is sufficient to generate the payoffs that each have a preference ordering on outcomes.) We can then look at the relative value of playing the various strategies in response to a given strategy on the part of the adversary. If we wish to look at the worst case, this is still represented in the game.

Another advantage is that we can generalize from the idea of computation by the legitimate users of the system and intrusion by adversaries. In our model we simply view the system as an engine for various outcomes based on the strategies of the various players. Who the 'good guys' are and who the 'bad guys' are can be tacked on as labels. But, primarily they are all just players and what counts is the desirability of the outcomes that their combined strategies produce.

If each node is a player with his own goals and strategies, the resulting game can be quite complicated. Typically, game-theoretic representation of n-person cooperative games involves looking at coalitions of players and side payments between them. (What is the best strategy for the coalition is not always best for individuals in it; the side payments guarantee that everyone in the coalition is best off if they fully cooperate with the rest.) However, we need not model things in this way. Suppose we can separate the nodes into 'good' nodes and 'evil' nodes. If the interests (represented by the payoff function) of all the good nodes coincide and those of all the evil nodes coincide, then the game can be reduced to a two-person game between the good network and the evil network. All the different actions possible for the different nodes of a single network become alternative moves by that network (player). For example, a node in a network sending a message is a move by the corresponding network/player, whether or not any other node in the network is ever aware of that move.

It is important to note the large potential advantage of being able to simplify to two-person games. Many results that apply in the two-person case have no generalization to the n-person case. Those that do generalize typically require difficult to meet or difficult to evaluate assumptions. Further, even when results exist, they can be prohibitively harder to compute in the n-person case. Still, even if payoffs coincide, the remaining problems may be far from trivial. In fact, even if we have a purely cooperative game between two players, if they cannot communicate, they might not be able to coordinate their strategies. For example, suppose that we have a simple game in which two players would like to cooperate. Their only strategy choice is to go either left (L) or right (R). They derive no benefit unless their choices coincide. We can represent this via the payoff matrix below.

	L	R
L	1 1	0 0
R	0 0	1 1

If the players can communicate, they can simply agree to go either right or left together. But, if they cannot, there is no way for them to coordinate. One approach that has been taken to this problem is to let the players play a repeated game. If a game is played repeatedly rather than just once, then players who cannot communicate can attempt to come to an equilibrium by adopting a strategy of varying their play and watching how other players vary their own play. Such games have generally been known as stochastic games or Markov games. Research into such learned coordination has been extensively studied in both artificial in-

telligence and game theory. In addition to being modelled as games, when payoffs coincide they may be represented as multiagent Markov decision processes. (A nice summary of these issues, which also relates games and multiagent Markov decision processes, is given in [2].)

We are obviously not interested in the purely cooperative case (except perhaps in modelling for nodes all in the same subnetwork). An interesting question is whether we are only interested in the purely competitive case, i.e., zero-sum games. If the goal of the evil network is pure denial of service, then this is almost certainly purely zero-sum. But, in most cases this is not so immediately clear. For example, if the goal of the evil network is to leak information discretely, this may require some actions that are helpful to the good network in other respects. The question is whether we can always represent the security relevant alternatives in such a case as a zero-sum game, leaving aside all the cooperative elements as irrelevant.

In our discussion so far we have been modelling the interests of individual nodes. Even when we reduce to a two-person game, we have been assuming that this is justified by a strict coincidence of interest by nodes of each subnetwork. But, each node need not be fully aware. We already observed that nodes of one player/network need not be aware of actions by other nodes of the same network. Further, it is not necessary to assume that the nodes of one network/player be aware which are the other nodes of that player. In fact, it is not even always necessary for a node to know whether it is good or evil itself. Games can be used to model behavior in which there is no rationality or intent by individuals at all. In evolutionary games, whole populations or species can be represented as players [11]. In fact, in this now long established application of game theory, even the rationality of the players may be only metaphorical. Rationality is a natural metaphor to the behavior that first motivated game-theoretic abstractions, and it is thus a natural concept for giving intuitive understanding to the mathematics. There is, however, nothing inherently requiring rationality in those mathematical abstractions. Though it may be surprising to the unfamiliar, we can similarly represent the opposing networks as players without considering the interests of individual nodes. This introduces yet another useful layer of abstraction in our model.

5. Summary and Future Work

Recently there has been a call to look at secure computing from the perspective of dependability, in which faults are expected to occur and the goal is to provide an acceptable degree of assurance even in the presence of such faults [12]. And, there have been some measures suggested for evaluating operational security in terms of reward to an attacker as a function of effort expended [9].

The above competing networks view proposes a more general model of computation in the face of hostile attack. It incorporates the perspective of hostile agents and friendly agents in a single computing framework with the same class of parameters affecting their goals. This allows us to characterize the risks and rewards of the competing agents in a single mathematical model, which we have suggested is naturally viewed game-theoretically. Thus, this paper amounts to a proposal for a computational and mathematical model in which to explore the dependability approach to secure computation. The topological observations above reflect an interesting abstraction in thinking about secure distributed computation. If we view all components of a network as either trusted or untrusted, then the entire network is two-colorable in this way (all connected components of the same color are treated as one node). This can quickly show us information relevant to what sort of distributed computation problems are potentially solvable by various (trusted or untrusted) parts of the network.

One natural direction to take the game theoretic aspects of this work is to apply the operational security measures proposed in [9] to determine the payoffs in the games of our model. We have also already posed a number of questions about the types of games relevant for our concerns. In addition to looking for broad answers to these questions we can look at answers tailored to specific operational issues. Another area for potential research is to look more closely at repeated play of games in the context where explicit agreement is not possible. What sorts of problems are solvable in competing networks and under what circumstances? Inasmuch as learning in such cases is a form of covert agreement, to what extent are the usual countermeasures to covert channels (e.g., [8]) relevant and in what way?

We showed that networks of n interacting nodes need not be represented by an n-person game: they can often be represented in a two-person game, which can greatly simplify analysis. Similarly, there need be no direct correspondence between the number of nodes in the actual system and the model. Specifically, we might have a standalone system for which we model the various processes as communicating agents. This is hardly new, but it means that our model need not apply only to networks. It can serve as a general model for secure computing. This is another potential avenue

of exploration.

Secure computing has typically been researched and developed against a backdrop of worst-case failure models. By weakening goals and assumptions about hostile failure we were able to derive a protocol that achieves a conditional form of probabilistic coordination even in the face of hostile attack. We weakened the usual byzantine failure model for secure distributed computation in both node behavior and connectivity. This produced a more realistic view of hostile failure. It also yielded a model that is more flexible both in pursuing solutions to existing problems and in pursuing (or indeed raising) new problems in secure distributed computation.

Acknowledgements

In addition to the anonymous referees, I would like to thank David Goldschlag, Steve Greenwald, and Cathy Meadows for helpful comments and discussions. This work was supported by ONR.

References

[1] Ross Anderson and Roger Needham. Programming Satan's Computer. In J. van Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 426–440. Springer-Verlag, Berlin, 1995.

[2] Craig Boutilier. Planning, Learning, and Coordination in Multiagent Decision Processes. In Yoav Shoham, editor, *Theoretical Aspects of Rationality and Knowledge: Proceedings of TARK VI*, pages 195–210, Renesse, Netherlands, March 1996. Morgan Kaufmann Publishers.

[3] Mathew K. Franklin and Michael K. Reiter. The Design and Implementation of a Secure Auction Service. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 2–14, Oakland, California, May 1995. IEEE Computer Society Press.

[4] Li Gong, Patrick Lincoln, and John Rushby. Byzantine Agreement with Authentication: Observations and Applications in Tolerating Hybrid and Link Faults. In *Preproceedings of DCCA-5: Fifth International Working Conference on Dependable Computing for Critical Applications*, pages 79–90, Urbana-Champaign, Illinois, September 1995.

[5] J. N. Gray. Notes on data base operating systems. In R. Bayer, R.M. Graham, and G. Seegmüller, editors, *Operating Systems: An Advanced Course*, volume 60 of *Lecture Notes in Computer Science*, chapter 3.F, pages 393–481. Springer-Verlag, 1978.

[6] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *JACM*, 37(3):549–587, July 1990

[7] Joseph Y. Halpern and Mark R. Tuttle. Knowledge, Probability, and Adversaries. *Journal of the ACM*, 40(4):917–962, September 1993.

[8] Myong Kang, Ira Moskowitz, and Daniel Lee. A Network Pump. *IEEE Transactions on Software Engineering*, 22(5):329–338, May 1996.

[9] Bev Littlewood, Sarah Brocklehurst, Norman Fenton, Peter Mellor, Stella Page, David Wright, John Dobson, John McDermid, and Dieter Gollmann. Towards Operational Measures of Computer Security. *Journal of Computer Security*, 2:211–229, 1993.

[10] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[11] J. Maynard-Smith. *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, 1982.

[12] Catherine A. Meadows. Applying the Dependability Paradigm to Computer Security. In *Proceedings of the New Security Paradigms Workshop*, pages 75–79, La Jolla, California, August 1995. IEEE Computer Society Press.

[13] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[14] Michael K. Reiter. A Secure Group Membership Protocol. In *Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 176–189, Oakland, California, May 1994. IEEE Computer Society Press.

[15] Michael K. Reiter. Secure Agreement Protocols: Reliable and Atomic Gorup Multicast in Rampart. In *Proceedings of the* 2^{nd} *ACM Conference on Computer and Communications Security*, pages 68–80, Fairfax, Virginia, November 1994. ACM Press.