Merging Paradigms of Survivability and Security: Stochastic Faults and Designed Faults *

J. McDermott
Naval Research Laboratory
Washington, DC 20375
USA
mcdermott@itd.nrl.navy.mil

A. Kim
Naval Research Laboratory
Washington, DC 20375
USA
kim@itd.nrl.navy.mil

J. Froscher
Naval Research Laboratory
Washington, DC 20375
USA
froscher@itd.nrl.navy.mil

ABSTRACT
Faults are examined by both the security and fault tolerance communities. These communities have strikingly different views of the types of faults that exist, the way they are modeled, and how they are addressed. One community can pronounce a system survivable but the other community would not find this to be so. This leaves us with two approaches that both fail to be comprehensive, depending on which community is looking at the system. While intrusion-tolerance and security researchers look at faults in terms of statistically dependent events caused by the hard intruder, the fault tolerance literature assumes that faults are statistically independent and can be described as random variables with probability distributions. When considering the survivability of a system, we cannot assume that the system is susceptible to only one type of fault or the other, but this is common practice in both communities. A new paradigm is needed.

1. INTRODUCTION
When thinking of survivable systems, we expect them to perform in the face of faults (or at least fail in the expected manner). Therefore, understanding, modeling, and correcting these faults are very important steps in the survivability arena. While system faults are examined by both the security and fault tolerance communities [7], those communities have strikingly different views of the types of faults that exist, the way they are modeled, and how they are addressed. The different communities can look at the same system and identify different sets of faults, thus also devising different survivability approaches. One community can pronounce a system survivable but the other community would not find this to be so. This leaves us with two approaches that both fail to be comprehensive, depending on which community is looking at the system.

*This work was produced by the United States Government and is not subject to copyright.

New Security Paradigms Workshop 2003 Ascona Switzerland
ACM 1-58113-880-6/04/04

Security researchers and fault-tolerance researchers look at survivability from opposing viewpoints. Security people view it in terms of trust relationships while the fault tolerance literature focuses on redundancy and reconfiguration In summary, one community models faults as worst-case behavior of hypothetical intruders while the other considers faults to be stochastic. This results in solutions from both paradigms that cannot handle faults from the other paradigm.

In this paper, we introduce some definitions and concepts that are important in understanding the conceptual differences between the two opposing literatures, describe the different types of fault classes and intruders that the two literatures focus on, and propose that a new paradigm shift is required in this area if a system is to be truly survivable.

1.1 Definitions
Powell, Stroud, et al.[13] provide an insightful interpretation of general dependability concepts [1, 9] for security. We follow their definition:

- *attack* - a malicious interaction fault aiming to intentionally violate one or more security properties; an intrusion attempt via a vulnerability.
- *vulnerability* - an accidental fault, or a malicious or non-malicious intentional fault, in the requirements, specification, design, implementation, or configuration of the system or its use, that could be exploited to create an intrusion.
- *intrusion* - a malicious, externally-induced fault resulting from a successful attack.

Following conventional security practice, we qualify attack, vulnerability, or intrusion with a general security property that may be violated: e.g. confidentiality, integrity, or availability. For example, we may have a confidentiality attack or an availability intrusion. This distinction is important because, for example, an approach that tolerates availability intrusions may not tolerate confidentiality intrusions. For example, redundant copies of a data item x allow a system to tolerate availability intrusions that damage some but not all of copies of x. However, a confidentiality intrusion that results in an unauthorized read of data item x cannot be tolerated by redundant copies, since the service (confidentiality of x) cannot continue, be restored, or be compensated for using the redundancy.

1.2 Hard Intruders and Gremlins

Security not only brings the notion of attack, vulnerability, and intrusion faults to dependability, it also brings with it the notion of an intruder. The significant characteristics of intruders are the rate at which they occur, their objectives, their capabilities, and their willingness to take risks. Of these characteristics, only the intruder's rate of occurrence is probabilistic and even then it is not ergodic.

In this paper we consider two kinds of intruders. In the spectrum of intruder characteristics these two represent extremes that make our point clear. Consideration of intruders, such as script kiddies, who fall between these extremes, obscures the point we are trying to make. One kind, *hard intruders*, have relatively high-value objectives, low risk aversion, high skills, and high resource levels. The other has no objective at all, low skills, low risk aversion, and the capability to attack any component at any point in its life cycle. We call the latter *gremlins*.

A hard intruder may be a team defending a world view (i.e. a very high value objective), some of the team members may have a very low risk aversion for this goal, the team may have many person-months to develop attack tools, and some team members may have high security experience. Our thesis and our experience is that hard intruders have a significant rate of occurrence for high-consequence systems. Since hard intruders have statistically dependent impacts on containment regions and components, Byzantine faults [12] do not model them accurately. We use the notion of hard intruders in a way that is analogous to Nielson and Nielson's hardest attacker [11]: we look at what are arguably the most difficult faults to address via fault-tolerance approaches.

In contrast to hard intruders we use the notion of spontaneous intruders or gremlins because they are arguably the most difficult to address via trusted approaches used to counter hard intruders. The term gremlin originated in the RAF during the first half of the twentieth century and referred to an imaginary gnome-like creature responsible for inexplicable failures in aircraft. Personify stochastic faults as gremlins to show how trusted, unbypassable, tamper-resistant components have difficulty in coping with stochastic faults. The most significant fact about gremlins is that they can attack any component at any point in its life cycle. Unlike hard intruders who are real persons, gremlins are imaginary beings that cannot be stopped by trusted design,development, and deployment. On the other hand, gremlins do not perpetrate very sophisticated attacks and have no specific objective. The damage or impact on one component is usually statistically independent of any impact on other components. Therefore, Byzantine faults can accurately model the behavior of gremlins.

2. PROBLEMATIC FAULTS

As aforementioned, the types of faults that are examined in the two opposing literatures can be categorized into two different classes. In a nutshell, the fault tolerance literature focuses on stochastic faults, and the security literature focuses on designed faults. Before we can address the different types of faults together, we need to examine each class of fault in more detail.

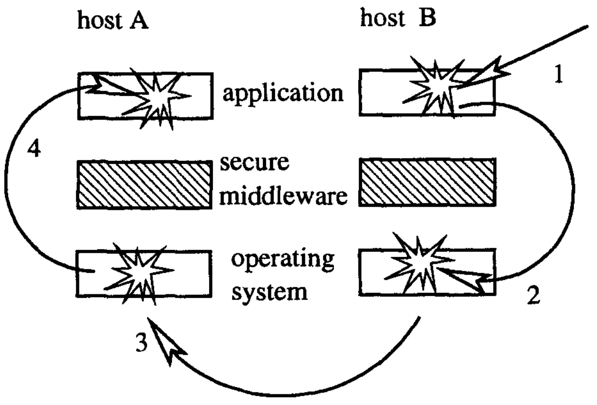

Figure 1: Architecture Attack

2.1 Designed Faults

Hard intruders cause *designed faults* [1]. According to our definition of an attack or intrusion as a fault, designed faults are attacks or intrusions that are matched to the design assumptions and assertions [2] about the system under attack. A designed fault invalidates one or more of the assertions or assumptions that intrusion-tolerance or security depends upon. Designed faults may include common mode faults as replicated attacks on redundant components, with the intention of defeating the redundancy. Designed faults may include architecture faults, as attacks or intrusions that are directed at a part of a system that does not directly enforce the policy being challenged. Architecture attacks or intrusions bypass protection mechanisms. For example, an integrity attack may be conducted via host operating systems when the applicable integrity policy is enforced by middleware, thus bypassing the defense.

Designed faults (attacks or intrusions) are overlooked by the fault tolerance community because they do not affect tolerance structures in statistically independent ways. Approaches based on redundancy only work if we assume that the attacks or intrusions are not replicated in a corresponding way. Approaches based on reconfiguration only work if we assume that the attack or intrusion does not reconfigure to match the new security posture. Designed attacks or intrusions can, by definition, be expected to employ the precisely corresponding techniques.

The limitation of fault-tolerance techniques is that they assume that random variables with tractable distributions accurately describe all faults. On the basis of these random variables, fault-tolerance approaches assume that some com-

[1]We mean "designed" and not "design."
[2]Assumptions are conditions on the environment of a system and assertions are conditions that the system satisfies.

ponents or configurations will not be affected by a fault. On the other hand, because the approaches assume [3] completely random behavior, they can deal with faults that occur in unpredictable locations with unpredictable behavior. The behavior of a designed fault is, from a fault-tolerance point of view, so unusual as to be practically impossible. Thus, no provision is made for dealing with designed faults. In fact, it would be awkward at best and intractable in most cases to try to model designed faults as random variables.

2.2 Stochastic Faults

Gremlins perpetrate stochastic faults. That is, there are no human sponsors behind the faulty behavior. Stochastic faults can be due to software flaws, hardware failures, unintentional misuse, or external damage such as fire or weather. Whatever the cause, the effect is the same as if imaginary but relatively ignorant persons were given unrestricted access to randomly chosen components.

Fault-tolerance approaches use redundant fault containment regions [3] to deal with stochastic faults. There is no attempt to reason about specific traces of behavior. Instead, some very general behavior such as fail-stop or Byzantine communication is assumed for the region as a whole, and the rest of the system is designed to operate with these kinds of faults in several of its regions. Because they make no assumptions about specific fault behavior, fault tolerance approaches are very powerful in the presence of stochastic failures. Trusted component-based approaches used by the security community on the other hand, find stochastic faults to be most problematic to deal with. Security approaches are intended to resist designed attacks and are based on models of hard intruders. A hard intruder is posed for each class of fault (e.g. confidentiality) and a careful design, development, and deployment process is followed. The goal of the process is a system comprising a (relatively) small number of trusted components with the rest being untrusted. The meaning of trusted is that 1) the hard intruder has no access to the trusted components and 2) hard intruder manipulation of any combination of untrusted components will not succeed, because of the way the trusted components interact. This trust is established by reasoning about sets of specific system traces and no random variables are used.

Trusted component approaches assume some components can be ruled inaccessible to intruders during some or all phases of their life cycle. Since gremlins can appear in any component, it is not possible to have a component that is trusted with respect to stochastic faults. Furthermore, since gremlins can exhibit a wide range of (stochastic) behavior, reasoning about a particular gremlin in terms of sets of traces is essentially intractable. The problem with these security approaches is that they assume that sets of traces describing the behavior of (possibly hard) intruders accurately models all faults.

From a trusted components point of view, gremlins (the intruders behind stochastic faults) are imaginary and thus not considered at all. Thus, no provision is made for dealing with them. No amount of logical verification can keep them out, because they are stochastic.

[3]From a certain point of view.

2.3 An Example

Govindavajhala and Appel show how soft memory errors can cause security flaws [4]. The immediate basis for the attack is a single upset event that flips a bit somewhere in memory. A carefully designed program can exploit this flipped bit, to bypass a type system used for language-based security[4]. The single event upset appears to be a stochastic fault, and by itself, it is. However, the attack described in this work is highly designed, that is, it is not easily modeled by a stochastic variable. The paper describes the designed nature of the attack quite well and includes an explicit discussion of the assumptions and assertions violated by the attack [5]. It might be easier to see that this is a designed fault if one recalls that controlled energy of some appropriate form is applied to the hardware, to exceed the level assumed for the physical environment of the hardware.

3. A PARADIGM SHIFT

The following table summarizes the major differences between the ways the two communities approach survivability in terms of faults.

These two problematic kinds of faults have limited the practical survivability of current and proposed survivable systems. Any survivable or intrusion-tolerant system that is based upon redundancy or reconfiguration and that does not consider hard intruders, is probably ineffective against designed attacks. Any survivable or intrusion-tolerant system that is based upon trusted, unbypassable, tamper-resistant components and that does not consider stochastic faults, is probably ineffective in the presence of gremlins. Current research in survivability and intrusion tolerance is proceeding in just this fashion. A paradigm shift is needed to build truly survivable systems.

There are at least three ways to shift toward the new paradigm: 1) from fault-tolerance approaches toward designed faults, 2) from trusted-component approaches toward stochastic faults, and 3) increasing the expressiveness of models such as stochastic process algebra [5] to encompass practical systems.

The first approach should be adopted when coming from the field of fault tolerance. Results should show the required trust relationships among redundant components of an intrusion-tolerant architecture and show how the redundant components can achieve the required level of trust. They should also seek to define significant hard intruders and show how the trust relationships frustrate these intruders.

The second approach should be the first step when coming from the security community. Results should be based on trusted component approaches but make provisions for dealing with stochastic faults through redundancy and reconfiguration. For example, multilevel secure database approaches could be adapted to make them Byzantine fault tolerant.

[4]It seems likely that many other security mechanisms could be bypassed by exploiting similar flaws.

[5]"All proofs of soundness are premised on the axiom that the computer faithfully executes its specified instruction set."

	Security Community	Fault Tolerance Community
Nature of Faults	Designed	Stochastic
Attacker	Hard Intruder	Gremlin
Approaches	Trusted Components	Redundancy and Reconfiguration
Weakness	Stochastic Faults	Designed Faults

Table 1: Characteristics of Problematic Faults from the Two Paradigms

Both approaches 1 and 2 can be applied with incremental extensions of known results from the appropriate community. However, both approaches 1 and 2 have the potential to merely shift the focus from one to the other without completely addressing the problems in each. Therefore, expanded models that encompass both types of faults (and intruders) are the ideal approach for dealing with the issues of stochastic faults and designed faults. Stochastic process algebra[6] is a good example of an expanded approach because it can model not only the functional behavior of concurrent systems, but probabilistic aspects as well, which are required when considering stochastic faults. For example, the mission of an organization (and the system that supports this mission) may be to deliver the correct computational results (functional) for a certain fraction of the time, given a certain rate of fault occurrence (probabilistic). Unfortunately, stochastic process algebras per se do not appear to be sufficiently well-developed for direct application to survivability. Further work is required by researchers in foundation issues, toward new expanded modeling approaches (e.g. improvements in stochastic process algebra).

3.1 An Example Paradigm

A simple application of stochastic process algebra will make the preceding discussion more concrete. We want to show two things with this example: 1) what a successful new paradigm might look like, and 2) the kinds of limitations that we find in current candidates for this paradigm.

We will use PEPA [6] as the stochastic process algebra, with some changes in notation that make security modeling easier. In PEPA the instantaneous *action* α of a conventional process algebra is replaced by the *activity* (α, r) where α is the *action type* and r is the *rate* of the activity. An activity (α, r) has a duration which is an exponentially distributed random variable. The rate r is the parameter for the distribution of the duration.

Our first extension is the use of *compound action types* for the *activities* of a process P. In basic PEPA, the action type of an activity is denoted either by a Greek letter such as α or an identifier such as *send*. For our purposes, we use compound action types where the components are composed by the ordered tuple notation, thus $\langle send, a, nonce_a\rangle$ represents the action type for sending a message containing Alice's identifier and a nonce. In PEPA a process X that engages in activity a of action type α with activity rate r and then acts like process P is denoted

$$X = (\alpha, r).P$$

or, with a compound action type

$$X = (\langle send, a, nonce_a\rangle, r).P$$

In addition to the change in notation we will also use renaming functions to establish associations between activities in different processes. For example, suppose we have two processes P_1 and P_2 defined as $P_1 = (send, r_1).P_1$ and $P_2 = (receive, r_2).P_2$. We wish to connect these two processes by arranging for their first activities to have a common activity type. This accomplished by a renaming function f defined as follows

$$\begin{aligned} f((send, r)) &= (receive, r) \\ f((\alpha, r)) &= (\alpha, r)\ ,\ \alpha \neq send \end{aligned}$$

When this function is applied to a process the result is a new process with the action types renamed according to the function. Using the function f defined above $f(P_1)$ becomes

$$f(P_1) = (receive, r_1).f(P_1)$$

We can now combine the two processes to communicate by means of the PEPA cooperation operator

$$f(P_1) \underset{\{receive\}}{\bowtie} P_2$$

The meaning of this construct is similar to the meaning of parallel operators in conventional process algebras. Activities in P_1 or P_2 with action types other than *receive* will proceed independently. Activities of type *receive* must complete in both P_1 and P_2, at the rate of the slower instance of *receive*.

The PEPA algebra was initially defined for performance modeling but we can apply it to model survivability in the presence of both designed and stochastic faults. PEPA models can be used as ordinary process algebra models, to show the effects of designed faults. To show the effects of stochastic faults we use a basic construction that starts by defining a constant process $FAIL$

$$FAIL \overset{def}{=} (\tau, \top).FAIL \qquad (1)$$

Process $FAIL$ only performs internal events with the unknown action type τ and don't care rate $\top$. We use process

[6]Stochastic Petri Nets (SPN) [10] are another possibility, but they do not model abstraction and composition as well as process algebras. It is difficult to compose a model of good components with an intruder model, using SPN. Another possible approach is the Box Calculus [2], an extension of Petri nets.

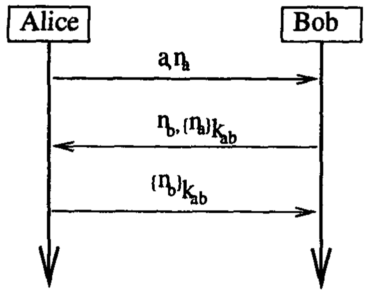

Figure 2: Protocol Sequence Diagram

$FAIL$ and the PEPA choice operator + to give every process the alternative of failing. For example, suppose we need to include a process $(\alpha, r).P$ in a model. To make this process fallible, we replace it with the process

$$(\alpha, \frac{(k-1)r}{k}).P + (\alpha, \frac{r}{k}).FAIL \tag{2}$$

This new process will perform an action of type α with rate r but then, with probability $1/k$, it may fail. (Our construction for fallible processes is reminiscent of transition-assigned-output state machines. Like the Mealy machine that must perform a transition to have an output, all fallible processes must complete at least one activity before failing.)

For our example, we will model a simple mutual authentication protocol taken from Kaufman, Perlman, and Speciner [8]. Alice wishes to establish a protected communications session with Bob. Alice starts the protocol run by sending her userid and a nonce to Bob. Bob responds with a nonce of his own and Alice's nonce encrypted with their shared key k_{ab}. Alice then confirms the session by responding to Bob with Bob's nonce encrypted with their shared key k_{ab}. The protocol steps are depicted in the sequence diagram of Figure 2.

We model infallible Alice [7] as the process shown in Equation 3. To simplify the exposition, we have shown each activity with the same rate r.

$$\begin{aligned} Alice = \ & (\langle send, a, n_a\rangle, r). \mathop{+}_{\substack{k \in Key \\ n \in Nonce}} \ ((\langle receive, n_b, \{n_a\}_{k_{ab}}\rangle, r). \\ & (\langle send, \{n_b\}_{k_{ab}}\rangle, r).Session(a, b, k_{ab}, n_a, n_b) \ ) \end{aligned} \tag{3}$$

The sub-process of receiving Bob's response, confirming Alice's identity, and running a session is modeled as a choice (+) indexed over all legal keys and nonces that Alice might encounter. The term $Session(a, b, k_{ab}, n_a, n_b)$ denotes a process that carries out a communication session using key k_{ab}, etc. We model infallible Bob in a similar fashion, with indexed choice used to model the fact that Bob is prepared to attempt a protocol run with any legal key and nonce.

$$\begin{aligned} Bob = \ & \mathop{+}_{\substack{k \in Key \\ n \in Nonce}} \ ((\langle receive, a, n_a\rangle, r).(\langle send, n_b, \{n_a\}_{k_{ab}}\rangle, r). \\ & (\langle receive, \{n_b\}_{k_{ab}}\rangle, r).Session(a, b, k_{ab}, n_a, n_b) \ ) \end{aligned} \tag{4}$$

By using our previously defined renaming function f, we can combine processes $Alice$ and Bob into a complete protocol run. This gives us a model of the protocol that is suitable for analysis wrt designed attacks.

$$f(Alice) \underset{\{receive\}}{\bowtie} f(Bob) \tag{5}$$

We can adapt the model of Equation 5 to look at stochastic faults by making $Alice$ and Bob fallible processes, using the approach of Equation 2. To simplify our exposition, we will assume that all activities occur at the same rate r and that all failures have the same probability $1/k$. A fallible $Alice$ is

$$\begin{aligned} Alice = \ & (\langle send, a.n_a\rangle, (k-1)r/k). \\ & \mathop{+}_{\substack{k \in Key \\ n \in Nonce}} \ ((\langle receive, n_b, \{n_a\}_{k_{ab}}\rangle, (k-1)r/k). \\ & (\langle send, \{n_b\}_{k_{ab}}\rangle, (k-1)r/k).Session(a, b, k_{ab}, n_a, n_b) \\ & \quad + \\ & \qquad (\langle send, \{n_b\}_{k_{ab}}\rangle, r/k).FAIL \\ & \quad + \\ & \qquad (\langle receive, n_b, \{n_a\}_{k_{ab}}\rangle, r/k).FAIL) \\ & + \\ & (\langle send, a, n_a\rangle, r/k).FAIL \end{aligned} \tag{6}$$

We also show a fallible Bob process as

$$\begin{aligned} Bob = \ & \mathop{+}_{\substack{k \in Key \\ n \in Nonce}} \ (\\ & (\langle receive, a, n_a\rangle, (k-1)r/k). \\ & (\langle send, n_b, \{n_a\}_{k_{ab}}\rangle, (k-1)r/k). \\ & (\langle receive, \{n_b\}_{k_{ab}}\rangle, (k-1)r/k).Session(a, b, k_{ab}, n_a, n_b) \\ & \quad + \\ & \qquad (\langle receive, \{n_b\}_{k_{ab}}\rangle, r).FAIL \\ & \quad + \\ & \qquad (\langle send, n_b, \{n_a\}_{k_{ab}}\rangle, r).FAIL \\ & + \\ & \qquad (\langle receive, a, n_a\rangle, r).FAIL \\ &) \end{aligned} \tag{7}$$

It should be clear at this point that stochastic process algebra can model both designed faults and stochastic faults. We can add an infallible intruder process $Yves$ to our system and demonstrate, via the process algebra itself, that $f(Alice) \underset{\{receive\}}{\bowtie} f(Bob)$ is susceptible to a designed attack[8].

[7]That is, we don't include failure probabilities using the method of Equation 2.

[8]Exercise for the reader: find the attack.

We can also derive an underlying Markov model from the same PEPA model. We will not present this derivation because it would detract from our example. It is sufficient to say that any finite PEPA model has a corresponding finite-state Markov process. The problem (and one of the foundational research issues) is that the Markov processes corresponding to PEPA models with failure have some states that are not *positive recurrent*[9]. The states corresponding to the process $FAIL$ constitute *absorbing boundaries* of the Markov process. Because of this, the process may not have a stationary probability distribution and if it does, the distribution may be difficult to find. Without a stationary probability distribution, it is hard to make concise statements about survivability wrt stochastic failures. So basic PEPA, while promising, is difficult to use as survivability paradigm.

4. CONCLUSIONS

Survivable systems need to not only correctly and accurately detect the presence of attack or intrusion faults, but also function properly (i.e. complete the mission) in face of these faults, especially in mission-critical systems. At the same time, these mission critical systems should also be able to survive faults that are random and unpredictable in nature.

Both intrusions and random faults are faults to the system, and should not be thought of separately when considering survivability of mission-critical systems. However, in reality, these two types of faults lack a common research plateau on which to define, model, examine, and counter faults. That is because, while both the intrusion-tolerance and fault tolerance communities examine system faults, these communities have strikingly different views of the types of faults that exist, the way they are modeled, and how they are addressed.

While intrusion-tolerance and security researchers look at faults in terms of statistically dependent events caused by the hard intruder, the fault tolerance literature assumes that faults are caused by gremlins and thus can be described as random variables with probability distributions. However, when considering the survivability of a system, we cannot assume that the system is susceptible to only one type of fault or the other.

For a system to be truly survivable, we must consider the failure behaviors of both classes of faults. In order to achieve this, we need to consider development of models based on a combination of stochastic behavior and the ability to reason about traces[10]. This kind of model can encompass both types of faults and methods of dealing with them. For this purpose we suggest a paradigm shift that enables research to merge these types of faults together.

This new paradigm would be much more useful since it can be used for all stages of assessing survivable systems including fault prediction, fault tolerance, fault recovery (removal), and validation. With these new research tools, we can design systems and support mechanisms that are tolerant against not only stochastic faults, but designed faults as well, creating a practical survivable system.

[9]A state X in a Markov process is positive recurrent if the expected number of transitions until the process returns to state X is finite.

[10]That is, specific detailed system behavior.

The position stated in this paper may appear obvious to the reader. Unfortunately, it is apparently not obvious to many of the researchers in the security and survivability communities. Both research communities have spent much time on complex algorithms or large prototypes that fail to address issues from the other discipline. Our approaches need to change.

The notion of "faults that are not easily modeled by stochastic approaches" raises an interesting possibility that there may be other significant classes of faults that are not stochastic faults but are also not designed faults. Designed faults invalidate the design assumptions or assertions of one or more survivability mechanisms. It could be possible that there are significant faults that do not violate design assumptions or assertions but are nevertheless not stochastic faults.

Acknowledgments

We would like to acknowledge the comments of the anonymous reviewers. Discussion with Bob Blakely, Sami Sayjdari, Mary Ellen Zurko, Steve Greenwald, and John McHugh resulted in improvements to this paper. We would also like to acknowledge B.B. Rath for posing the problem that led to our results.

5. REFERENCES

[1] A. Avizenis, J. Laprie, and B. Randell. Fundamental concepts of dependability. In *Third Information Survivability Workshop*, Boston, MA, October 2000.

[2] E. Best, R. Devillers, and J.G. Hall. The Box calculus: a new causal algebra with multi-level communication. In *Advances in Petri Nets*, volume 609. LNCS, 1992.

[3] C. Davies. Recovery semantics for a db/dc system. In *Proc. ACM Annual Conference*, pages 136–141. ACM Press, 1973.

[4] S. Govindavajhala and A. Appel. Using memory errors to attack a virtual machine. In *Symposium on Security and Privacy*. IEEE, May 2003.

[5] H. Hermanns, J.-P. Katoen, J. Mayer-Kayser, and M. Siegle. Towards model checking stochastic process algebra. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *2nd Int. Conf. on Integrated Formal Methods (IFM2000)*, 2000.

[6] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[7] Pankaj Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall PTR, 1994.

[8] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, 1995.

[9] J.-C. Laprie, J. Arlat, J.-P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J.-C. Fabre, H. Guillermain, M. Kâniche, K. Kanoun, C. Mazet, D. Powell, C. Rabéjac, and P. Thévenod. *Dependability Guidebook*. Cépaduès-Editions, Toulouse, 1995.

[10] M.K.Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on Computers*, 31(9):913–917, September 1982.

[11] H. Nielson and F. Nielson. Hardest attackers. In *Proc. Workshop on Issues in Theoretical Security*, Geneva, July 2000.

[12] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JACM*, 27(2):228–234, April 1980.

[13] D. Powell and R. Stroud. Malicious- and accidental-fault tolerance for internet applications: Conceptual model and architecture. Technical report, MAFTIA deliverable D2 (available as LAAS-CNRS Rep. 01426 or University of Newcastle upon Tyne CS-TR-749), November 2001.