Science of Cyber-Security

Contact: D, McMorrow - dmcmorrow@mitre.org

November 2010

JSR-10-102

JASON
The MITRE Corporation
7515 Colshire Drive
McLean, Virginia 22102-7508
(703) 983-6997

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
November 19, 2010	Technical	

4. TITLE AND SUBTITLE

Science of Cyber-Security

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER
13109022

5e. TASK NUMBER
PS

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

The MITRE Corporation
JASON Program Office
7515 Colshire Drive
McLean, Virginia 22102

8. PERFORMING ORGANIZATION REPORT NUMBER

JSR-10-102

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

ODUSD (AT&L)/RD /IS
1777 North Kent Street, Suite 9030
Rosslyn, Virginia 22209

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved or public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

JASON was requested by the DoD to examine the theory and practice of cyber-security, and evaluate whether there are underlying fundamental principles that would make it possible to adopt a more scientific approach, identify what is needed in creating a science of cyber-security, and recommend specific ways in which scientific methods can be applied. Our study identified several sub-fields of computer science that are specifically relevant and also provides some recommendations on further developing the science of cyber-security.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Steven E. King
a. REPORT	b. ABSTRACT	c. THIS PAGE			**19b. TELEPHONE NUMBER** *(include area code)*
UNCL	UNCL	UNCL	UL		703-588-7414

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Contents

Abstract

JASON was requested by the DoD to examine the theory and practice of cyber-security, and evaluate whether there are underlying fundamental principles that would make it possible to adopt a more scientific approach, identify what is needed in creating a science of cyber-security, and recommend specific ways in which scientific methods can be applied. Our study identified several sub-fields of computer science that are specifically relevant and also provides some recommendations on further developing the science of cyber-security.

1 EXECUTIVE SUMMARY

The need to secure computational infrastructure has become significant in all areas including those of relevance to the DOD and the intelligence community. Owing to the level of interconnection and interdependency of modern computing systems, the possibility exists that critical functions can be seriously degraded by exploiting security flaws. While the level of effort expended in securing networks and computers is significant, current approaches in this area overly rely on empiricism and are viewed to have had only limited success.

JASON was requested by the DoD to examine the theory and practice of cyber-security, and evaluate whether there are underlying fundamental principles that would make it possible to adopt a more scientific approach, identify what is needed in creating a science of cyber-security, and recommend specific ways in which scientific methods can be applied.

The challenge in defining a science of cyber-security derives from the peculiar aspects of the field. The "universe" of cyber-security is an artificially constructed environment that is only weakly tied to the physical universe. Therefore, there are few *a priori* constraints on either the attackers or the defenders. Most importantly, the threats associated with cyber-security are dynamic in that the nature and agenda of adversaries is continually changing and the type of attacks encountered evolve over time, partly in response to defensive actions. For this reason, no one area of science (mathematical, physical, or social) covers all the salient issues. Nevertheless, there are analogies with other research fields. Cyber-security requires understanding of computer science concepts, but also shares aspects of sciences such as epidemiology, economics, and clinical medicine; all these analo-

gies are helpful in providing research directions.

Our study identified several sub-fields of computer science that are specifically relevant. These include model checking, cryptography, randomization, and type theory. In model checking, one develops a specification of an algorithm and then attempts to validate various assertions about the correctness of that specification under the specific assumptions about the model. Model checking provides a useful and rigorous framework for examining security issues. Cryptography, which examines communication in the presence of an adversary and in which the assumed power of that adversary must be clearly specified is viewed today as a rigorous field, and the approaches pursued in this area hold useful lessons for a future science of cyber-security. Type theory is any of several formal systems that can serve as alternatives to naive set theory and is also effective in reasoning about the security of programs. The use of obfuscation, in which one attempts to disguise or randomize the data paths and variables of a program, can help in constructing defenses against some common modes of attack. Finally, game theoretic ideas will be useful in understanding how to prioritize cyber defense activities. It is not possible to protect everything all the time and so some notion of risk must be established; game theoretic approaches provide a framework for reasoning about such choices.

The development of metrics associated with the level of security of a computer system is clearly desirable, but it is important to understand and appreciate their limitations. It is certainly possible to chronicle various existing attack strategies and ensure that a system is not vulnerable to these, but this is at best a backward-looking approach. Change detection of files and key programs can help in identifying anomalies, but correlating such anomalies to actual attacks will require

further research using ideas originating in fields such as machine learning and event processing.

JASON identifies a need to accelerate the transformation of research results into tools that can be readily used by developers. There are some very sophisticated approaches (model checking, type checking etc. as discussed previously) that can be used to assess and reason about the security of current systems, but they are not widely available today in the form of developer tools. There may be an insufficient market for private development of such tools and this may argue for a more activist role on the part of DOD in supporting future development.

DOD posed a number of questions as part of the study. We quote below these questions and our responses.

1. *What elements of scientific theory, experimentation, and/or practice should the cyber security research community adopt to make significant progress in the field? How will this benefit the community? Are there philosophical underpinnings of science that the cyber security research community should adopt?*

   The most important attributes would be the construction of a common language and a set of basic concepts about which the security community can develop a shared understanding. Since cyber-security is science in the presence of adversaries, these objects will change over time, but a common language and agreed-upon experimental protocols will facilitate the testing of hypotheses and validation of concepts. The community can benefit here if there emerges a consensus on progress as well as more concrete notions of what future directions are most promising. At the same time, there must be

a connection with practice "in the wild" (in the same sense as one progresses from animal model tests to possible clinical trials in medical research).

2. *Are there "laws of nature" in cyberspace that can form the basis of scientific inquiry in the field of cyber security? Are there mathematical abstractions or theoretical constructs that should be considered?*

   There are no intrinsic "laws of nature" for cyber-security as there are, for example, in physics, chemistry or biology. Cyber-security is essentially an applied science that is informed by the mathematical constructs of computer science such as theory of automata, complexity, and mathematical logic.

3. *Are there metrics that can be used to measure with repeatable results the cyber security status of a system, of a network, of a mission? Can measurement theory or practice be expanded to improve our ability to quantify cyber security?*

   There are many metrics that could be productively employed, such as those that inform modern intrusion detection systems. But it must be understood that any such metrics are empirically based and statistical in nature; they cannot apply to scenarios that are not well defined. In particular, things that are not observed such as new attack approaches are not going to contribute to metrics. It is not possible to definitively measure a level of security as it applies to general operation of information systems. The repeatability of results hinges upon the imposition of community protocol standards and adherence to these criteria. At present, however, repeatability is not a central criterion for publication of results in cyber-security.

4. *How should a scientific basis for cyber security research be organized? Are the traditional domains of experimental and theoretical inquiry valid in cy-*

ber security? Are there analytic and methodological approaches that can help? What are they?

There is every reason to believe that the traditional domains of experimental and theoretical inquiry apply to the study of cyber-security. The highest priority should be assigned to establishing research protocols to enable reproducible experiments. These should provide clear descriptions of the initial conditions, the types of allowable threats and clear meanings for the security goals.

5. *Are there traditional scientific domains and methods such as complexity theory, physics, theory of dynamical systems, network topology, formal methods, mathematics, social sciences etc. that can contribute to a science of cyber security?*

   There are several areas that are traditionally a part of computer science that have contributed in the past to a deeper understanding of cyber-security and where increased future emphasis could bring continued improvement. The field of model checking seems particularly relevant in that one creates a model for the security of a given system or key kernel and then tests the assumptions using a well defined set of possible inputs. While the input space is potentially infinite, the benefit is that specific threats can be modeled. The area of cryptography has traditionally focused on provable aspects of secure communication with careful attention paid to the nature of assumptions. The application of code obfuscation and type theory also provide important insights into the construction of secure code. Finally, there is an important role for game theoretic approaches to security evaluation. For DOD, there is also value in exploiting secrecy as a factor in cyber defense. Examples include use of obfuscation as described above, development of

DOD-specific security products and in general collecting and securing data about cyber attack that are not publicly shared.

6. *How can modeling and simulation methods contribute to a science of cyber security?*

   There are a number of ways in which modeling and simulation can contribute. One is by using existing computational capability to continually test security assumptions on running systems. Another is to use concepts of virtual machines, etc. to provide well-defined test beds to explore in a controlled way the behavior of computational and security systems in the presence of well-defined attacks.

7. *Repeatable cyber experiments are possible in small closed and controlled conditions but can they be scaled up to produce repeatable results on the entire Internet? To the subset of the Internet that support DoD and the IC?*

   It is premature to ask this question, as there would appear to be little organization of the results of previous small scale experiments. As many of these were not performed with the goal of repeatability, it is important to first assess the utility of these smaller scale test beds before contemplating the scale-up to controlled wide area networks or to the Internet in general.

8. *What steps are recommended to develop and nurture scientific inquiry into forming a science of cyber security field? What is needed to establish the cyber security science community?*

   The establishment of interdisciplinary centers that connect academia, industry, national laboratories and the DOD would be an important step in this direction. Such centers should focus on cyber-security issues of particular relevance to DOD needs, but would also leverage the activities of other

important contributing agencies such as DARPA and the Information Assurance efforts within the NSA. In all such enterprises it will be important to establish protocols developed for, and peer-reviewed by, the community so as to facilitate the communication and archiving of results.

9. *Is there reason to believe the above goals are, in principle, not achievable and if so, why not?*

   There is every reason to expect that significant progress can be made toward the above goals. One must first understand the nature of the scientific enterprise for cyber-security and characterize the objects under discussion. There is a great deal of valuable science that can be accomplished if an accepted approach to discourse can be developed. While these primarily technical activities will not in and of themselves "solve" the cyber-security problem given that it has both technical and social aspects, they will significantly aid progress.

2 PROBLEM STATEMENT AND INTRODUCTION

JASON was tasked by the DOD to perform a study on the interplay of science with cyber-security. An excerpt from our charge follows verbatim:

> "The Department of Defense, the Intelligence Community, and the planet have become critically dependent on the Internet for services, transactions, social interactions, communications, medical treatments, warfare; virtually everything. Cyber-security is now critical to our survival but as a field of research does not have a firm scientific basis. Clearly, there is a scientific basis for the infrastructure of the internet such as computation, communications, and integrated circuits but its security attributes are often vague or un-measurable. Small changes at the bit level can have large and often poorly understood results for end-node or network security. There are concerns that future damage could be catastrophic to major components of our core infrastructures such as power and water distribution, finance and banking, even the ability to deploy forces to defend the country.
>
> Our current security approaches have had limited success and have become an arms race with our adversaries. In order to achieve security breakthroughs we need a more fundamental understanding of the science of cyber-security. However, we do not even have the fundamental concepts, principles, mathematical constructs, or tools to reliably predict or even measure cyber-security. It is currently difficult to determine the qualitative impact of changing the cyber-infrastructure (more secure now or less secure?) much less quan-

> tify the improvement on some specific scale. We do not have the tools to do experiments that can produce results that could be compared to theory, models, or simulations. Request the JASONs consider whether cyber-security can become or should be a science. If so,identify what is needed to create a science of cyber-security and recommend specific ways in which scientific methods can be applied to cyber-security. If not, what can we learn from the practice of science that would enable us to improve the security of our cyber infrastructure and assure the integrity of information that resides in the information technology infrastructure?"

There is general agreement that there are security problems in cyber-space. For instance,

> "Internet security problems are becoming more conspicuous with each passing day. Online information such as pornography and obscenities are seriously harming the physical and mental health of minors. Criminal activities such as online fraud and theft are seriously harming public security. Computer viruses and hacker attacks are posing serious threats to the security of the operation of the Internet. Leaking of secrets via the Internet is posing serious threats to national security and interests."

This quote is from an April 29, 2010 speech by Wang Chen, the chief internet officer of China entitled *On the development and management of the internet in our country* A U. S. official probably would not have emphasized pornography,

but otherwise the problem is accurately described. (The approach the Chinese are taking wouldn't be acceptable here either.)

In this report, we summarize the findings of our study on the science of cyber-security. Because cyber-security is a large field that benefits from developments in a number of areas such as computer science, mathematics, economics and social science it is impossible to be complete as regards all the topics that were presented to us.

Our report is organized as follows. In Section 3 we discuss our overall findings and conclusions regarding the role of science in cyber-security. This discussion attempts to draw some parallels with other scientific areas, and also attempts to identify where those areas may be of value for future research. We then provide longer discussions on two key areas that were felt to be particularly relevant. The first, discussed in Section 4, is the role of logic-based tools that can actually verify various propositions regarding the security of a given program. While this area is very promising, there remains much to be done in making it easily usable for today's complex systems. The second, discussed in Section 5 considers the use of immunology as a potential candidate for the science of cyber-security. We draw some connections but conclude that making this analogy is as yet premature. Finally, in Section 6 we summarize our overall conclusions regarding the role of science in cyber-security.

3 CYBER-SECURITY AS SCIENCE – An Overview

D. Stokes, in *Pasteur's Quadrant* [18], suggests a tripartite division of research, based on whether the research is purely a quest for understanding, purely based on considerations of use, or both. He calls these "Bohr's quadrant", "Edison's quadrant", and "Pasteur's quadrant" respectively. It is possible to do good research in cyber-security purely in a quest for understanding. Likewise, it is possible to do good research in cyber-security aimed at developing solutions to immediate problems. Our conclusion is that the science of cyber-security should lie firmly in Pasteur's quadrant. Cyber-security requires both a deep understanding of underlying principles, and close contact with developing issues. It is not necessary that any one research group be equally good at both of these, or even specific topics from both.

Supporting this position is not a wholly comfortable place to be. It invites criticism on two fronts. Working on a deep understanding of underlying principles does not help with the difficult and urgent problems faced today. Likewise, research aimed at immediate problems frequently falls short of solving those problems, and may well be evanescent, leaving behind no particularly useful results. Each individual is likely more in one camp than the other, giving greater value to understanding or to applicability. Indeed, this is true for the people named on the title page of this report. The report attempts to show that Pasteur's quadrant is the right place to be, by discussing other sciences with some of the characteristics of cyber-security, and by mentioning a range of examples.

There are no surprises in this report, nor any particularly deep insights. Most people familiar with the field will find the main points familiar. There may be errors

in the report, and substantive disagreements with it, but during the preparation of this report, mostly we heard about differences of emphasis. Some people occupied other points on the Bohr-Pasteur-Edison spectrum, and some felt that there were many sciences of cyber-security, but that it wasn't one science. We think these positions are consistent with our view of the field.

3.1 Attributes for Cyber-Security

A science of cyber-security has to deal with a combination of peculiar features that are shared by no other area of study. First, the background on which events occur is almost completely created by humans and is digital. That is, people built all the pieces. One might have thought that computers, their software, and networks were therefore completely understandable. The truth is that the cyber-universe is complex well beyond anyone's understanding and exhibits behavior that no one predicted, and sometimes can't even be explained well. On the positive side, the cyber-universe can be thought of as reduced to the 0s and 1s of binary data. Actions in this universe consist of sequences of changes to binary data, interleaved in time, and having some sort of locations in space. One can speculate as to why mathematics is so effective in explaining physics, but the cyber-world is inherently mathematical. Mathematics is a natural way for reasoning about it and this point of view is a theme that appears repeatedly in this report. Second, cyber-security has good guys and bad guys. It is a field that has developed because people have discovered how to do things that other people disapprove of, and that break what is thought to be an agreed-upon social contract in the material world. That is, in cyber-security there are adversaries, and the adversaries are purposeful and intelligent.

Thus, the reasoning in cyber-security has to be both about the constructed universe and the actions and reactions of the adversaries. Definitions and concepts will shift over time in cyber-security, as in all sciences. As quoted in [15]

> A successfully chosen name is a bridge between scientific knowledge and common sense, between new experience and old habits. The conceptual foundation of any science consists of a complicated network of names of things, names of ideas, and names of names. It evolves itself, and its projections on reality changes.

This quote echoes the observations of a presentation to JASON by Peter Gallison of Harvard in which he compared the type of science encountered in cyber-security to a "Manichaean science" or science in the presence of adversaries. Particularly important for this type of science (and indeed any branch of science) was the development of a common language so that precise arguments could be formulated that addressed directly the problems of interest. We emphasize this point because the definition of cyber-security itself is imprecise. The subject includes correctness properties like confidentiality and integrity, and liveness properties such as are associated with denial-of-service attacks, together with a growing set of unpleasant surprises.

3.2 Guidance from other Sciences

In this section we consider some other sciences in the hope of extracting some indications of what might be expected as the science of cyber-security develops. We emphasize sciences that resemble cyber-security, for instance with active agents,

or those in which things change over time. It is unlikely that the science of cyber-security will look much like the universal truths of general relativity and we conclude that because of its overall artificial construction, there are no "fundamental" laws of cyber-security as one has for example in physics

3.2.1 Economics

Economics is a social science explicitly concerned with competing agents. Economics has a deep mathematical component, and is routinely used to make more or less specific predictions. Economic models span the range from detailed and data-heavy to completely unrealistic but useful for insight [19]. As everyone knows, many of the predictions are more like tendencies, and the record of success is mixed. Nevertheless, economics plays an important role in that economic considerations can shape the expectations of adversaries.

3.2.2 Meteorology

Meteorology is based entirely on physics. For a while it was hoped that by using models that incorporated more physics and by gathering more data it would be possible to make high-accuracy forecasts far into the future. Although forecasting has become more accurate and somewhat longer-range, the naive hope is not realizable. In fact here, the issue of the limits of predictability are as important as the fundamental description of various physical processes. In addition, it is well known that there are extreme events that are very difficult to predict with accuracy but their existence requires that plans be put into place to deal with their consequences.

3.2.3 Medicine

Medicine today is primarily organized experience with science-based tools. The basic science of how cells work, or how the body reacts to specific molecules, is still lacking. There are no fully quantitative measures of health, although there are quantitative test results. Nor can the health of individuals be compared; one cannot say "I am 6 dB (or Pasteurs) healthier than you." Nonetheless, where modern medicine is available, health has improved beyond the dreams of our ancestors. One class of basic objects, diseases are described in a generally useful way. Some have been eliminated completely, many are treated fairly successfully. Despite all the effort (including the $30,000,000,000 spent by the United States on research annually) some diseases, like influenza, are only managed, killing tens of thousands each year. The aspirations of the field are bounded by experience; no one foresees providing immortality.

3.2.4 Astronomy

Astronomy is an observational science. Scientists do not do astronomical experiments. Of course the fields that underlie astronomy, physics and chemistry, have strong experimental foundations. In addition, the universe, as far as we know, is not adversarial. We believe that if well-posed questions are asked, the responses are repeatable within the context of the underlying physics.

3.2.5 Agriculture

Agriculture, or at least those aspects that match our topic, has crops that are grown, and pests that attack the crops. Over time, crops can be modified to make them more resistant to pests. More commonly, the pests are attacked directly, by pesticides, or by biological control. Without continual attention (or even with it) the pests come back.

We believe the attributes given above are useful in understanding how the science of cyber-security will develop and what it will look like, at least for the foreseeable future. Note that it is applied science and engineering that deals with problems in the real world. The science doesn't solve the problem, but may well enable the solution.

3.3 Security Degrades Over Time

We observed earlier that success in cyber-security requires constant attention. Not only do the threats and attacks change, but the old techniques tend to become less effective. Technology advances, and the properties of the security system become better understood. Below are two examples:

3.3.1 Unix passwords

The original UNIX passwords were up to 8 ASCII non-control characters. The passwords aren't stored directly. Rather hashes (some one-way function) are stored. In modern systems these hashes aren't easily read, but if a machine is

compromised, then the hashes can be taken away, and one can attempt to break passwords by making up likely candidates, applying the one-way function, and seeing if the result matches any of the hashes. There are about 2^{53} original UNIX passwords, which seemed like quite a lot at the time. Nowadays, one could imagine trying about 32 million guesses (2^{25}) per second on a single CPU core, so testing the entire space would only take about 3 days on 1,000 cores. That's not a particularly large computer, but the calculation is also embarrassingly parallel, so the 1,000 cores could be 1,000 machines in a (small) botnet. Or one could use 9,000 machines, still not a large botnet, and do the whole calculation in one night. For completeness we note that there has been a lot of work on making it harder to guess passwords; for instance, by using much slower one-way functions and much larger spaces. Another approach is to force people to use more complex passwords, but that has other sociological ramifications which are unfortunately very difficult to deal with in terms of any science. An extreme version turned up in the recent Russian spy case, where

> "Ricci said the steganographic program was activated by pressing control-alt-E and then typing in a 27-character password, which the FBI found written down on a piece of paper during one of its searches."

3.3.2 Lock bumping

A second example of security decreasing over time comes from the common locks people use on doors. A simple technique called "lock bumping" makes all these locks quite easy to open with just a little practice. The locks were, of course, vulnerable, but it didn't much matter until the technique was popularized starting

about 2005. There are plenty of explanatory videos online.

The implication for cyber-security is that the nature of the attack can change over time. It will not in general be possible to anticipate all attacks. For this reason it becomes necessary to constantly survey the extant modes of attack to understand the nature of current technology as well as continually attack one's own systems in the hopes of finding heretofore undiscovered attacks.

3.4 The Role of Secrecy

All security regimes include some secrets. The TSA doesn't explain everything it is looking for; the credit card processors don't fully explain their criteria, and so on. In the computer science community, security by obscurity is generally considered bad, probably based on examples of flawed cryptography. There seems to be little theoretical work on this subject. We mention a few examples of keeping (and not keeping) secrets.

Commercial anti-virus programs, of the sort that run on your home computer, reveal some of the weaknesses of totally public defenses. In principal, a malware author could just buy these, and adjust the malware until none of them detect it. The market has made this easier; there are online services that for a fee will run uploaded malware through a large number of commercial programs, and report the result. (One example is at `scan4you.biz`.)

Game theory and economics teach us that keeping our private valuations secret sometimes adds value. Much effort has gone into auction mechanisms that get bidders to reveal their private valuations. On the other hand, under various circum-

stances, this "asymmetric information" can lead to lemon markets, where either no market exists, or only people who are forced to will trade in the market.

Secrecy might also be required by the rules. For instance, the DoD might use logs of network traffic to tune its network defenses. These logs might easily contain material that should not be made public. For these reasons, there may be some benefit in exploring the role an enhanced role for secrecy and obfuscation, particularly for DOD systems.

Finally, one might ask if keeping the source code secret adds much to security. First, there may be other reasons for keeping source code secret, such as the trade secrets incorporated in it. Certainly attackers with source code are likely to proceed differently, as they have a better chance of finding attacks based on logical errors, and this would be especially valuable for targeted attacks. On the other hand, there seems to be no particular shortage of effective attacks when source code is unavailable. But the code owner may have no realistic alternatives to making the source available. As an example consider the following announcement:

> "The agreement that we signed with the FSB is an extension of Microsoft's Government Security Program (GSP)," Microsoft said in a statement on Friday. "The purpose of the GSP is to increase trust with national governments. In the case of the Russian agreement, GSP participation will facilitate the development of the next generation of secured solutions for Russian government agencies based on the latest Microsoft technologies and Russian cryptography."

3.5 Aspects of the Science of Cyber-Security

It is not simple to define what the "security" in cyber-security means. Precise definitions are necessary, but all too often it is "That shouldn't have happened." For instance, when you insert a USB memory stick into a computer, and the system consequently runs some program you've never heard of, that's bad. But when you start your computer, the system runs lots of programs you've never heard of, and generally that's not bad. Calling the first program "unauthorized" and the second sort "authorized" is an imprecise beginning. And this raises an important issue with cyber-security. Precise definitions matter. Until there is a precise set of objects that can examined carefully and clearly, it will not be possible to increase the level of rigor. An example of areas where precise definitions are key is the study of cryptography and the use of model checking which we discuss in more detail in Section 4

By analogy with medicine or agriculture, we expect no magic bullets. Success, however measured, will require constant attention. Cyber-space will remain attractive to criminals, spies, vandals, and other bad folk, and they will inventively press for advantage. On the other hand, human adversaries can sometimes be deterred or dissuaded, which is not possible with disease agents. For many years, banks had layers of annoying physical security, bullet proof glass, armed guards, and sometimes locked doors, all to deal with bank robbers. Banks don't look like that any more. They have taken steps so that the robbers have mostly gone elsewhere such as limiting cash on hand in small branch offices.

3.6 Some Science

We give some illustrations or work, or potential areas for more work, that illustrate our idea of the science of cyber-security. For most of these there is an attempt to describe how the work is having, or might have, an impact on today's cyber-security problems.

3.6.1 Trust

Trust is fundamental in internet commerce, or in even knowing what to think about a piece of email. It is unclear how to think about trust or to model its ebb and flow. Is there some sort of Second Law of Thermodynamics of trust, where trust starts high and is dissipated over time? Or is it the contrary, that trust starts low and can grow through a series of good experiences? Is it more complex, and how can the waxing and waning be thought about? Careful definitions again seem crucial, together with the evaluation of the strength and reach of the resulting theorems.

3.6.2 Cryptography

Cryptography has produced many interesting ideas and important applications. We mention only one of these, the idea that it might be possible to search for malware signatures without revealing what we're searching for, or whether we found them. The idea would be to use some form of homomorphic encryption to carry out a computation on the suspected malware, and return, from each such computation, an encrypted version of the result. Were such a scheme devised, it would have the property that the adversary cannot see what's going on, without a huge

amount of computation. The point of such a scheme follows from our discussion of secrecy. As long as the adversary cannot observe what we do with the answer, the adversaries cannot tell if we think their code is malware. The existence of a practical way of doing this is, as far as we know, still an open problem. However, there has been progress in this area as briefed to us by Rivest [17].

3.6.3 Game theory

Game theory has been used successfully in several areas. The approach explicitly models the interests of attackers and defenders and their repeated interactions. We mention an interesting study on the use of game theory in understanding wireless bandwidth allocation strategies [4] in which it is used to find and prove properties of protocols for ameliorating the impact of cheaters in ad hoc 802.11 networks. Similar work is described in [2].

The simplest version of a game is a two-player game that is played once. Each player has a set of actions, and knows what the value of each action will be, but the value depends on what the other player does. So in principle each player considers all its options, based on what the other player might do, contemplating all its options. Each player adopts a *strategy*, possibly involving random choices, describing what they will do. There is more than one optimality concept. A *Nash* equilibrium occurs when neither player can do better just by changing their strategy. The two strategies together are *Pareto* optimal when every change that makes one player better off harms the other player. (This is usually expressed as the contra-positive: a change is Pareto-better when one player is better off and the other is no worse off. It is Pareto optimal when there is no Pareto-better strategy.) There can be multiple Nash equilibria, and multiple Pareto equilibria, and they

need not agree.

A more complicated concept of a game is relevant in wireless networks, because things happen over and over, and there are typically more parties involved. Multiple parties only make the analysis more complicated, but repeated games add many subtleties which are well beyond our scope, so we merely outline what can happen in a case much simpler than the ad hoc 802.11 network mentioned above.

Consider the simple Forwarder's Dilemma game, in which player A scores 1 if player B forwards A's message (at a cost to B of ε for power), and vice versa. So if A decides to forward and B does not, A's payoff is $-\varepsilon$ and B's is 1. One round of this game looks like

	Forward	*Deny*
Forward	$(1-\varepsilon, 1-\varepsilon)$	$(-\varepsilon, 1)$
Deny	$(1, -\varepsilon)$	$(0,0)$

(3-1)

DD (both Denying to forward) is a Nash equilibrium, as it is for any finite repetition where the players know when the game ends. (In the last round they will both Deny, and knowing that, do the same in the move before, and so forth.) But in a series of repetitions where the players don't know when the game will end (even with discounting the future) there are other, more socially desirable Nash (and Pareto) equilibria. For instance, if a player Forwards as long as the other does, and then Denies after that, the total payoff to the first player that doesn't Forward goes down. To the extent that this payoff models the players' values, each will continue forwarding the other player's packets, because of the existence of such an equilibrium.

3.6.4 Model checking

Model checking is a powerful tool that has traditionally been used for uncovering problems in communication protocols. However, it can be used to understand any program including those associated with security protocols. The basic idea is that one describes a system in some formal language. Then for each desirable property the system is supposed to have, the model checker looks for a counterexample to $SYSTEM \implies PROPERTY$. We use the phrase "model checking" expansively, including for instance Alloy (`http://alloy.mit.edu`), whose authors explain how it differs from state space searchers based on temporal logics. For us the common feature is being able to assert that various properties hold and then using these tools to construct counterexamples. More details of the power of this approach are provided in Section 4.

3.6.5 Obfuscation

Obfuscation of various sorts can help both the attacker and the defender, especially in low-level attacks. An article by Erlingsson [6] is a detailed, but clear, introduction to the topic. The idea for defenders is that if the memory layout of programs can't be reliably observed by attackers, then there is a whole class of attacks that become more difficult [20]. All the major operating systems do some amount of this, although generally there isn't enough room in 32-bit systems to do a good job. Nor do application writers always use the available tools. This technique is but one facet of a general question, whether the defenders can change (or appear to change) more quickly than the attackers. There are lots of approaches that might be studied, including various forms of randomization, multiple imple-

mentations, or rapid and frequent restarts from a checkpointed state. There are more than engineering questions here. How do the proposed approaches compare with each other, which attacks do they affect, and which do they not, and indeed, under what circumstances does one approach subsume another?

3.6.6 Machine learning

One tool for recognizing attacks involves machine learning. The rapid spread of viruses like Code Red some years ago led to epidemiological-style modeling that showed that waiting until an attack was generally recognized made it impossible to stop the virus from spreading (see the discussion at `http://ccied.sysnet.ucsd.edu/tutorial`). One result has been an arms race in which defenders attempt to characterize properties of the network representation of malware, and attackers attempt to look more and more like normal network traffic. Malware comes in many forms, and it is hard to measure the effectiveness of malware detection (as an example of how hard security metrics are in general). The same techniques, however, can be used on spam, where effectiveness seems easier to understand. Some big US internet mail providers, Google, Microsoft, Yahoo, use user feedback and machine learning to help classify emails. Access to huge amounts of traffic helps them, and the filters have become very good. Unfortunately, decent continuing comparative studies seem hard to find.

3.6.7 Composition of components

Is it possible to build secure systems out of insecure components? More realistically for the short term, is it possible to combine less secure systems to get

more secure systems, and if so, what are the comparative strengths and weaknesses of each approach? Approaches to the problem are not limited to cyber-security. Auditing and double-entry bookkeeping are designed to control both fraud and mistakes. Various forms of auditing have been proposed to improve the security of electronic voting over that offered by single computer based systems. It is much the same to have information flow through multiple paths and check that the results are identical. For instance, imagine running multiple firewalls or virus checkers in parallel, and only allowing traffic that every one agreed was acceptable. (In this simple scheme false alarm rates go up too.) There are many examples of this sort of approach to ensure the resulting system is more nearly correct. As an example, in AT&T's 5ESS switch there was a process that scanned through memory looking for and fixing data structures it thought were corrupted. This process and what it did were not error-free, but it did make the system much more reliable. Indeed the whole set of security-improving features of a modern operating system have some of the desired character. It is unlikely that any one of them is implemented perfectly, but in combination they add a lot to the security of the system. On the other hand, just piling on mediocre security patches can't be the right answer. The operating system mechanisms tend to have some amount of orthogonality. As usual, measuring the effectiveness of the result is hard, but it is not so hard to figure out which attacks are defeated, which made more difficult, and which left unaffected.

3.7 Applying the Fruits of Science

Even now there are many research results connected with cyber-security. These get applied when the research community engages directly, for instance in study-

ing internet standards and security-related protocols. However, there are far more areas where security is critical than the research community could handle. Many of these expose the same issues over and over, adding no research value.

If the science is to have impact, then it is necessary to bring the fruits of research to a larger community. This is a familiar issue in many areas of technology. The government frequently uses the technical readiness level (TRL) to measure how mature a technology is, and how suitable it is for use in a system. TRL 3 is about when research has shown feasibility, and TRL7 is about when the technology could be (adventurously) put into systems, or used by those who live on the bleeding edge. There is a big difference in usability between these two levels, which isn't always recognized by the people involved, be they researchers or those who fund research.

Being at TRL 7 is not necessarily sufficient for a successful commercial product since it is not always clear that there is enough of a market to support the whole process of refinement and adaptation to produce commercial products and tools. A striking example of some of the issues is given by Coverity. Coverity is a static analysis tool for C and C++ programs. It attempts to identify data races, memory corruption, and the like by analyzing source code. For instance, it would attempt to make sure that when the program executes, every lock that is acquired is released exactly once or insuring that a variable would not be referenced after freeing its memory. In a fascinating paper (see [3]) the authors describe the obstacles and where their effort went. Particularly striking was the effort it took to adapt to their users' environments, an activity necessary for commercial viability, but irrelevant to the research. Like most such programs it does not attempt to show that a program is bug free. Also, and this is an important issue in getting people

to use them, static checkers tend to have false positives, reporting potential bugs that the users don't care about. Either users can show that the bug cannot actually happen, or they don't care about it, for instance it is on a code path involved in shutting down. The first kind of false positive occurs because static checkers are weaker than those model checkers that produce examples of failures.

We touch on two of the difficulties they describe. First, they have to find the code to check it. The customers tended to have automated build systems of many sorts, and there was no obvious place to plug in their checker. Various alternatives led them to intercepting every system call during the build so they could find the user's source files. Second, they had to parse the program. As they say,

> "Parsing is considered a solved problem. Unfortunately, this view is naive, rooted in the widely believed myth that programming languages exist."

Adapting to the enormous variety of non-standard dialects took a huge amount of work to get what they describe as full version-specific bug compatibility.)

Model checkers are a specific area where, we believe, more usable tools would lead to improvements. Two examples are Spin (`http://spinroot.com`) and Alloy (`http://alloy.mit.edu`). These are free tools with excellent web sites including tutorials and many examples. Yet tools like this seem to be used mostly by the research community, and are certainly not pervasive.

In aerospace, the rule of thumb is that each increase in TRL level costs another factor of two or three. We don't know of corresponding data in software, but a factor of two is certainly plausible. Getting the fruits of research has been messy and uncertain. We have no particular advice, other than to say that the process

needs to be supported somehow. It will not happen by accident.

3.8 Metrics

Measurement is generally seen as fundamental to progress in science. From that, it is a short step to looking for security metrics that quantify the security state of a system. We have a few observations on this subject.

First, there are many statistics out of which such a measure could be built. For instance, one could count, possibly with weights, successful attacks. After all, not all detection is done in real time. One could measure how fast recovery is. One could measure peculiar network traffic, both inside one's network, and at the boundaries. One could measure inexplicable system failures, for some will be hardware, some bugs, and some security related. One could quantify the results of audits, for instance of unknown executables, or of the rate of change of files, or of possible security-affecting events in system logs. One could attempt to measure dubious user behavior. One could measure patch levels, both operating system and application. Out of these and other statistics it ought to be possible to produce two sorts of security estimates: the extent to which the system is protected against known attacks and how well they are dealt with, and the frequency of event which might have some security implications. This is an engineering challenge.

Second, the metrics have to change over time, to match the changing environment. Attacks change, defenses change, new weaknesses are discovered. The metrics need to adjust, requiring on-going effort. It would also be useful to keep the old metrics, to try to understand trends.

Third, you can't measure what you don't observe. The implication is that even the best metrics can't measure everything. This is familiar from medicine. People who don't have, or complain about, symptoms are only tested periodically, and the tests are for the most common problems.

3.9 The Opportunities of New Technologies

One of the difficulties of cyber-security is that it has to be done in a rapidly changing environment. New technologies provide new services for users and new opportunities for attackers. Therefore there will be new chores for defenders. The science of cyber-security will have to accommodate an expanding world.

Here are some examples:

Cell phones By now cell phones are commonplace, but cell phones that are also hand held computers are spreading rapidly. These encourage users to load and use a wide range of applications, including banking and email. Thus far (summer of 2010) their different operating systems, or some other factor we don't understand, have protected them from wide-ranging attacks. In the absence of deeper understanding, there is no reason to believe this desirable state will last.

Radios Similar in concept, but with different vulnerabilities, are IP-based radios that organize themselves into ad hoc networks. The Army has worked on these, and armies exist because of adversaries. Ad hoc networks are likely to be fragile, so it is easy to imagine various kinds of denial-of-service attacks on these networks.

Smart grid The smart grid is coming, with computers for controlling power in individual houses, and perhaps sending messages upstream to the power companies. The latter have an incentive to make sure customers don't give themselves free power. There are broader worries about the security of the system and its behavior under attack.

Multi-core CPU's On a completely different note, most CPUs will be multi-core, and the existence of multiple cores provides opportunities for attackers and defenders. As an example of a potential vulnerability, consider one way kernels protect themselves from bad user data. The kernel checks the addresses and lengths of buffers being passed to it, and then either reads or writes them. Another core in the same address space has a brief interval between the checking and the use of the buffer to change the checked and accepted values into exploitable values. Again, model checking approaches as described in Section 4 can detect such problems.

Cloud computing Cloud computing may be upon us, even if no one knows exactly what the definition is. But if one imagines a cloud as being many virtualized servers, then it may be relatively easy to observe them, and restart ones that seem to be behaving badly, including those that show signs of being infected by malware.

There is talk about redoing the Internet because security of the existing one seems so hard. To some extent this is happening, not at the IP layer, but at the application layer. Web services brought to users through browsers are proving to be an opportunity for security, which is a good thing considering how much bad stuff comes over the web. In particular, browsers act like operating systems in this world, but ones that can be upgraded frequently. They are virtual machines (HTML5,

Javascript, etc.) whose behavior is for the most part based on standards. They are still malleable, but the backwards-compatibility problems are comparatively moderate. Thus approaches to security might be possible here that would be too radical to be successfully applied to traditional operating systems. Further, because there are standards, formal techniques can be used to study security, see for instance [1]. Among the results described are a simple new HTML5 form vulnerability involving a cross-origin redirect.

3.10 Experiments and Data

Data are crucial to scientific progress. A valuable caricature of how science progresses is that better observations lead to more probing questions which leads to better theory, and the cycle repeats. How might the science of cyber-security fit into this picture?

We note (observe?) that there are many sciences which are primarily observational rather than experimental. Among these are astronomy, epidemiology, demography, and to a large extent, economics. Much of the experience underlying existing research in cyber-security is observational.

The crucial feature that data must have, be it observational or experimental, is that it be generalizable. Experience from our networks or our systems is surely valuable for us to understand what's going on in our world, but it advances the science only to the extent that someone can figure out what it means in more general environments.

In any case there is a hunger for experimental results. This can be seen in what

gets reported at computer science conferences, although many computer scientists, including several of our briefers, bemoan the low standards for doing and reporting experiments. DARPA has been supporting an ambitious project for a National Cyber Range that is to be capable of supporting realistic environments along with comprehensive data collection. The hope is that the results will be experiments with reproducible results. Whether this will be worthwhile remains to be seen. It would have been better to have had results from smaller test ranges first, to guide the development of the bigger one.

The objects of study in biology are enormously complex, yet the biologists have found ways of doing useful experiments. We're not claiming there is any particular analogy. Our observation is that complexity alone is not always a barrier to effective experimentation.

In medicine, where the situation is complicated by ethical considerations, there are a lot of observations, as well as clinical studies of varying quality. There is an organization, The Cochrane Library (`www.cochrane.org`) that produces expert reviews of the literature on specific topics. We suspect cyber-security would benefit from a similar approach. To start with, it would be worth seeing review papers that take an objective look at various topics. In particular, the proposers of various improvements are always enthusiastic about the benefits, but most proposals have a range where their impact is positive, and a range of similar-sounding problems where their impact is neutral, at best. Public appraisals of strengths and weaknesses would add to the science.

4 MODEL CHECKING

In this section we discuss the role of model checking and formal methods in computer security. In this approach, a designer must first create a formal specification of the algorithm under construction including its communication pattern. Model checking then can be used to simulate the control flow and either verify or refute various assertions about the code such as the impossibility of reaching various states or the absence of deadlock etc. Traditionally, this approach has been used for various types of asynchronous protocols to ensure that the protocols function reliably. Today, however, many codes are asynchronous by nature either because they use threading or distributed computing and so model checking has become valuable in this arena as well.

The work described in this section largely follows a presentation provided to JASON by Dr. Gerard Holzmann [9] who has developed Promela (PROcess MEta LAnguage) and a verification and simulation engine called Spin (Simple Promela INtepreter) [10]. We begin by describing some of the essential aspects of the language and then describe an application to the well-known Needham-Schroeder key exchange protocol. We then discuss further the promise of model checking for more complex applications. We close with a discussion of generalizations of these ideas such as the use of hyper-properties as developed by Schneider [5]. Our purpose here is not necessarily to endorse the use of Spin for analyses of cyber-security but to point out that model checking is by now a very mature area where the basic language of analysis has been determined (the language of linear temporal logic) and the objects associated with verification are quite precisely

```
active proctype main ()
{
  printf("hello world\n")
}
```

Figure 4-1: The "hello world" example expressed in the Promela language [10]

stated. It is these attributes which define a scientific approach, and the adoption of these types of well-defined concepts would be of benefit in future work on cyber-security.

4.1 Brief Introduction to Spin and Promela

Like all languages there seems to be an obligation to provide an example that types "Hello World" as in the first exposition of the C language. In Promela this looks as in Figure 4-1.

While this looks like any programming language there is a fundamental difference. Promela is a process modeling language. There is no floating point or other support for data types more complex than integers or strings. In the case above a process called `main` is created and then prints out a string.

A more complex example, the traditional producer-consumer makes this more evident. Consider the listing in Figure 4-2. This code defines two processes, producer and consumer and three global enumeration variables, P, C and turn. When the code is executed, the two processes are instantiated and then examine the global variable turn to determine what to do. There is no notion of time ordering here. One cannot assume either process started before the other. The "do" loop in each process repeats indefinitely. Within the loop a guard statement (turn ==P) is ex-

```
mtype = {P, C}

mtype turn = P;

active proctype producer()
{
  do
  :: (turn == P) ->
  printf (``Produce\n'');
  turn = C
  od
}

active proctype consumer ()
{
  do
  :: (turn == C) ->
  printf (``Consume\n'');
  turn = P
  od
}
```

Figure 4-2: A producer-consumer example in Promela [10]

amined and the print statement can only execute if the guard is true. If it is not true the process will in this case block until the guard is true which provides a natural way to express process synchronization.

The above example is quite trivial, but it is easy to construct examples where the presence of multiple asynchronous processes that share variables can make it very challenging to verify that a program is behaving properly. Spin provides a very sophisticated verification approach which can check assertions (e.g. this variable never went negative) as well as investigate claims of safety (the code does not do certain bad things like corrupt certain variables etc.) and liveness (the code performs the functions that were intended).

```
byte cnt;
byte x, y, z;
active [2] proctype user()
{       byte me = _pid + 1;      /* me is 1 or 2 */
L1:     x = me;
L2:     if
        :: (y != 0 && y != me) -> goto L1
        :: (y == 0 || y == me)
        fi;
L3:     z = me;
L4:     if
        :: (x != me)  -> goto L1
        :: (x == me)
        fi;
L5:     y = me;
L6:     if
        :: (z != me) -> goto L1
        :: (z == me)
        fi;

L7:     /* success: enter critical section */
        cnt++;
        assert(cnt == 1);
        cnt--;
        goto L1
}
```

Figure 4-3: A flawed version of Dekker's algorithm for mutual exclusion expressed in Promela [10]

This is better illustrated by the code in Figure 4-3. The example is a flawed approach to Dekker's algorithm where two processes are instantiated and one process attempts to access a "critical section" of code without the interference of the other process. The two processes attempt to examine global variables to see when it is safe to take a counter (cnt), add one to it, and then subtract one from it. Because of the asynchronous nature of the two processes the challenge is to create a condition where the other process knows not to try to modify the cnt variable. It

is actually quite hard to reason about this in the absence of any guarantee on time ordering.

The Spin system has a verification mode which can check if there is an execution path that will exemplify the flawed nature of this version of the algorithm. The verifier attempts to falsify that the variable cnt is 1 when we assert it is. Indeed there is a time line of processes that will produce such a state and Spin can find it using a number of search strategies.

Using the verifier, we can see that the following output shows a counter example to the assertion that the variable `cnt` has the correct value of 1.

It can be seen that while process 0 was attempting to perform the critical section, process 1 incremented the counter and violated the assertion. Such an error is quite difficult to spot and the search capabilities of Spin are far better at doing this than a human. It is possible to fix this flawed algorithm by adding some extra information that coordinates among the processes and informs a given process whether it is its turn to enter the critical section. The original algorithm by Dekker corrects this problem. In this corrected version the verifier is unable to find a situation in which the assertion fails.

The Promela language also has other constructs which allow one to specify communicating asynchronous processes; this is invaluable in checking program correctness. One disadvantage, however, is that Promela analyzes program *specifications* and not whole programs. We will return to this point at the end of this section. We next consider the application of Promela to computer security.

```
Starting user with pid 0
Starting user with pid 1
  1:    proc  1 (user) line   5 ...[x = me]
  2:    proc  1 (user) line   7 ...[(((y==0)||(y==me)))]
  3:    proc  1 (user) line   9 ...[z = me]
  4:    proc  1 (user) line  11 ...[((x==me))]
  5:    proc  0 (user) line   5 ...[x = me]
  6:    proc  0 (user) line   7 ...[(((y==0)||(y==me)))]
  7:    proc  1 (user) line  13 ...[y = me]
  8:    proc  1 (user) line  15 ...[((z==me))]
  9:    proc  1 (user) line  19 ...[cnt = (cnt+1)]
 10:    proc  0 (user) line   9 ...[z = me]
 11:    proc  0 (user) line  11 ...[((x==me))]
 12:    proc  0 (user) line  13 ...[y = me]
 13:    proc  0 (user) line  15 ...[((z==me))]
 14:    proc  0 (user) line  19 ...[cnt = (cnt+1)]
spin: line  20 "mutex_flaw.pml", Error: assertion violated
spin: text of failed assertion: assert((cnt==1))
 15:    proc  1 (user) line  20 ...[assert((cnt==1))]
spin: trail ends after 15 steps
#processes: 2
                cnt = 2
                x = 1
                y = 1
                z = 1
 15:    proc  1 (user) line  21 "mutex_flaw.pml" (state 21)
 15:    proc  0 (user) line  20 "mutex_flaw.pml" (state 20)
2 processes created
```

Figure 4-4: A producer-consumer example in Promela

4.2 Application to Security

An important component of computer security is the use of modern cryptographic protocols. While many of these are, in principle, quite simple to state, their implementation is notoriously difficult because of complicated side conditions associated with regular use of a computer system that are generally never considered

when a secure protocol is verified. These side conditions lead to the possibility of attack and, because enumerating all such attacks is very difficult, if not impossible, the "real world" analysis of protocols becomes very important.

In a paper by P. Maggi and R. Sisto [14], the use of the Spin model checker is investigated in verifying the famous Needham-Schroeder protocol exchange algorithm. In particular, Spin is used to examine the set of communication patterns used by the protocol subject to the assumption that there is an intruder that can monitor the communication.

4.2.1 The Needham-Schroeder Protocol

The Needham-Schroeder protocol was developed in 1978 [16] as a way of initiating authenticated secure communication between an initiator A, a responder B.

The exchange protocol is shown graphically in Figure 4-5. The protocol uses public key cryptography. Each participant, the initiator and responder denoted collectively by H , possess a public key denoted in the figure by $PK(H)$ as well as a secret key $SK(H)$. Messages encrypted with the public key can only be decrypted via the secret key. The secret key for participant H is known only to H while the public key is available to any participant and in this case we will assume it has been posted to a key server S.

In order to communicate with H the sender encrypts a message x using the public key to produce an encrypted message $xPK(H)$. Only a receiver (hopefully only H) that has $SK(H)$ can decrypt the message. In addition this approach has the

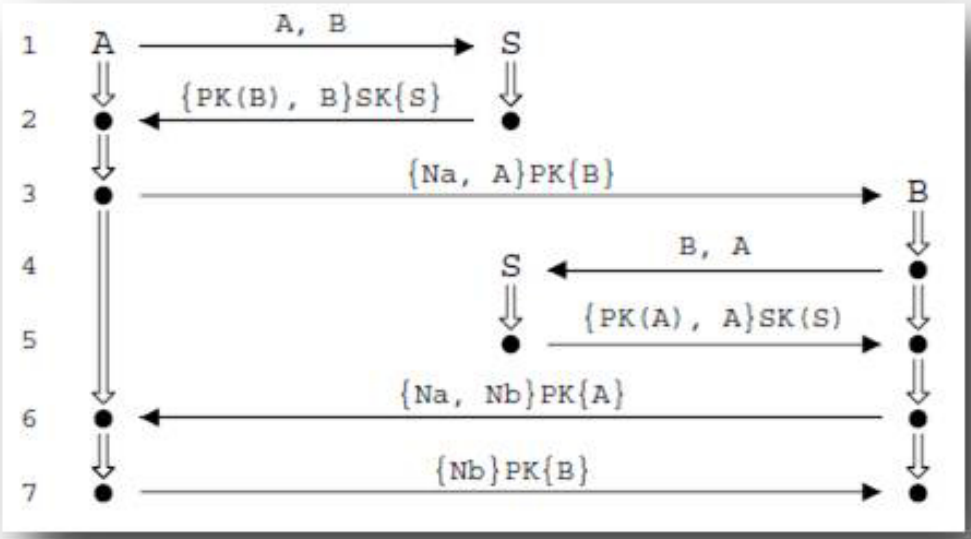

Figure 4-5: The Needham-Schroeder Key exchange protocol

nice feature that one can verify a message came from H if it is encrypted with the secret key owned by H; such a message can only be decrypted with the public key $PK(H)$. Thus the approach provides the ability to encrypt and it also provides the ability to apply a secure signature to authenticate the messages.

The protocol involves seven steps as shown in the Figure. Here A is the initiator, B is the responder, and S is the key server.

Step 1 A requests B's public key from the server S.

Step 2 The server S sends the public key but uses its secret key to encrypt it so A will know it came from S and if S is trusted, then this is really B's public key. Note that we assume here that the public keys are not compromised

but we can make sure the keys are fresh by using time stamps.

Step 3 Once *A* has *B*'s public key, *A* selects a nonce, a random number that will be used just once in the protocol sequence called *Na*. *A* sends the nonce to *B* using *B*'s public key. *B* decrypts the message from *A* .

Step 4 *B* then requests *A*'s public key from the key server *S*

Step 5 The server securely sends back *A*'s public key.

Step 6 *B* then sends its own nonce *Nb* as well as *A*'s nonce using the public key it got from the key server.

Step 7 *A* responds with *B*'s nonce encrypted with *B*'s public key.

At the end of this exchange both parties have their respective public keys and have proven to each other that they can encrypt messages using those private keys and have presumably authenticated themselves. In fact, if *A* and *B* already knew their respective public key then there is no need for the server and in that case steps 3, 6 and 7 make up the protocol.

4.2.2 Promela model of the protocol

In essence, the protocol exchange is very simple if there is no trusted server required. In Promela, this is expressed as shown in Figure 4-6. The initiator of the protocol exchange is labeled as an autonomous process as is the responder. There is also a process that models the intruder which we discuss below. What is not shown in the listing is that there are some additional labels that are transported as part of the message streams among the processes that indicate which process

```
mtype = {A, B, I, Na, Nb, gD, R};

active proctype A_initiator()
{       mtype g1;
        if
        :: party = B
        :: party = I
        fi;
        ca !  A, Na, A, party;
        ca ? A, Na, g1, A;
        cb !  A, g1, party
}

active proctype B_responder ()
{       mtype g1, g2, g3;

        ca ? g1, g2, g3, B;
        ca !  g1, g2, Nb, g3;
        cb ? eval(g1), Nb, B
}
```

Figure 4-6: The Promela specification of the Needham-Schroeder exchange

is sending or receiving a message. A brief examination of the code in Figure 4-6 shows the basic exchange for the initiator expressed in terms of communicating channels. Promela has a syntax for communication on channels that is modeled after Hoare's Communicating Sequential Processes (CSP). A channel can either send (denoted by !) or receive (denoted by ?). We can see then that the initiator sends out its encrypted nonce, gets back its nonce and the encrypted nonce of B and sends back B's nonce.

The intruder code is shown in Figure 4-7. The intruder process is assumed to be listening to all messages and can also inject messages into the stream. The complex "do" statement works as follows. The statements preceded by :: are option sequences. If only one of the options can be executed then it will be and

the loop starts again. If none of the options is executable the loop will block until one becomes executable. If more than one of the options are executable then some fair choice is made among the possibilities. It is this latter aspect which allows the possibility of intercepting messages at some point without interrupting the normal flow of messages among the initiator and responder. Executability in our case means that a channel is ready to be read or is ready to accept a transmitted message.

The intruder attempts to interject itself so as to learn various aspects of the exchange. The intruder also understands the nature of the exchange and has its own public and secret key (denoted by $PK(I)$ and $SK(I)$). A list of the things the intruder can learn from the various messages is shown in Figure 4-8. A list of the things the intruder needs to obtain is shown in Figure 4-9.

Using this specification in Promela it is possible then to verify the following assertion: when responder B commits to a session, the session was initiated by initiator A and not the intruder I. The way in which Spin verifies this is to see if it is ever possible within the state space of possible program traces to satisfy the contrary. If one such case can be found then clearly the assertion is not true. The correctness requirement is expressed using linear temporal logic and this is then translated into a criterion that can be checked against a running Promela specification.

The implementation in Spin produces the output shown in Figure 4-10. The Spin analyzer quickly finds a trace in which the initiator ends up committing to the intruder rather than the responder. In addition Spin provides the details of the trace that led to this violation. The interaction in this case is shown in Figure 4-11. The problem here is that the protocol as it was expressed is vulnerable to a

"man in the middle" attack in which the intruder masquerades as the responder B and the initiator A. This problem had been understood earlier by Lowe [13] but in this case it was determined through a computational analysis. The algorithm can be fixed in several ways, but the main attractive point for this study is the use of an "adversarial" tool to analyze an important security component.

There has been much controversy about the above example with the main argument that it is overly simplistic and does not reflect current practice. But in fact such simple interactions are the basis of more complex protocols. Consider, for example, the CCITT X.509 protocol for mutual authentication and confidentiality. It can be used to establish session keys. The idea here is similar, two processes wish to authenticate each other so they can exchange messages. A diagram of the protocol is shown in Figure 4-12. Here again we have an initiator (A), a responder (B) and an intruder (C). The protocol has only three messages. In the figure, T_a and T_b are time stamps, N_a and N_b are nonces, and X_a, X_b, Y_a, and Y_b are user data. Again we're using public key cryptography as the mechanism for encryption and signing with K_a and K_a^{-1} representing A's public and private keys and likewise for user B.

As discussed by Josang [11], the protocol as described is also vulnerable to a replay or "man in the middle" attack. Again the intruder C manages to make B believe that it is talking to A and so the authentication aspect of the protocol is violated. The problem here is that the users don't check their nonces and it is possible to replay an old nonce. By encrypting more information it is possible to avoid this problem. The problem outlined here can be exacerbated when multiple sessions of the protocol are initiated.

A specification of this protocol is constructed by Holzmann and one can ask for example if the number of exchanges say for two sessions is correct (in this case six exchanges). A verification of the Promela specification shows that an intruder can successfully masquerade as one of the users, but the number of exchanges goes up beyond the expected value of 6 when the exchange is performed properly and this can be checked. One set of messages that shows this and which was discovered as part of the search process is displayed in Figure 4-13.

4.3 Scaling Issues

The Spin system uses the notion of linear temporal logic to express the types of properties one wants to demonstrate or refute. Each claim written in linear temporal logic can be translated into an automaton or finite state machine that only accepts those executions that satisfy the logic constraint. In order to verify a property, one specifies the system and its "language" - the set of runs that the system can generate. One then defines the language of the property automaton. To prove that the system satisfies the property, one wants to show that every element of the system's language is also in the set of the language of the property to be checked. An equivalent way to express this is that the intersection of the language of the system and the complement of the language of the property is the empty set. Showing that the set is empty generally requires a search procedure to exhaust the possibilities. However, the violation of the property requires the existence of only one counterexample.

It turns out that the worst case computational expense of verifying any type of correctness property will increase with the number of reachable states of a model.

Holzmann makes the following order of magnitude estimate. Suppose we have n concurrent components (processes) and let m be the number of data objects they access. Suppose process i has T_i control states and the number of possible values associated with the j'th data object is D_j. Then in the worst possible case the total number of states in the model will be

$$\prod_{i=1}^{n} T_i \prod_{j=1}^{m} D_j, \tag{4-2}$$

so that the state space can grow exponentially with the number of concurrent components and data objects.

As an example, consider the operation of a message buffer. Suppose we have q buffers and let s be the maximum number of messages one can store in an individual buffer and let m be the maximum number of different message types. Then the total number of states is seen to be

$$R = \left[\sum_{i=0}^{s} m^i \right]^q . \tag{4-3}$$

The behavior of this formula is shown in Figure 4-14. As shown in the Figure, if the number of messages per buffer and the number of buffers are fixed then as we increase the number of types of messages the complexity grows geometrically. If we fix the number of message types and number of buffers then as we increase the number of messages per buffer the complexity is exponential. But if we increase the number of messages and number of buffers together while keeping the number of message types constant we get a double exponential growth and the number of states to check can increase astronomically.

For this reason, the use of a model checker has to be done in a very judicious way. Holzmann discusses the need for abstraction and restriction in the construction of

the model to tame the size of the search space. Even so, a complex algorithm can still produce a very large search space.

In order to cope with this, several approaches have been put forward with some degree of effectiveness. One approach is to exploit equivalences among classes of traces. For example slicing methods attempt to eliminate redundant data or control structures along the lines of what an optimizing compiler will do. Partial order reduction is an approach which attempts to identify redundant searches through the space where, for example, the result of the execution of two processes is independent of their order of execution. These types of optimizations can provide exponential reduction of state space.

Another clever approach is to use a type of lossy compression to avoid visiting previously seen states. Each state is stored in a state space which can get quite large. If a state descriptor takes 1kByte of storage we could store perhaps 10^7 states in a 32 Gbyte memory. In many cases, one would like to avoid visiting states that were already seen. To enable a fast look-up, Holzmann describes the use of Bloom filters. Bloom filters are very effective is determining whether a state has been seen before. Each state is hashed N times and if it appears in the table it has been seen before. If it has not been seen before, the hash is added to the table. As the number of hashes used to encode the states increases, the probability of a hash collision in which a state that has never been seen but hashes to a number that is in the table decreases exponentially. This makes it possible to consider large problems provided one is willing to deal with the occasional false positive.

Finally, the search procedure is essentially embarrassingly parallel making it an

ideal use for clusters of computers. Here the idea would be to use the innate advantage of the DoD in high performance computing to check code with more complex functionality. The common wisdom about model checking is that only very small code kernels can be checked this way, but using some of the ideas discussed above it is possible to verify some fairly complex code. For example, we were briefed on the use of Spin to verify the working of a real time operating system developed originally by Honeywell called DEOS. The DEOS system is used in various embedded systems. There are approximately 22 million states that must be traversed with about 584 bytes per state. An exhaustive search would require 13 GBytes. But by using 100 processors in parallel with a randomized approach to the state space search each process required only 8 Megabytes and the entire calculation achieved 100% coverage of the state space.

4.4 Extracting Models from Code

Given the power of this approach, it is natural to ask whether one can subject code "as is" to a Spin verification. In general this will always require some sort of assistance from the analyst to develop some abstraction of the verification required. Holzmann provides an interesting example of an attempt to verify the simple "Fahrenheit to Celsius converter" described in the original text on the C language by Kernighan and Ritchie. The source for this program looks as in Figure 4-15.

There are several issues in converting this simple code. First, Promela does not have a floating point data type and also does not support a while loop. But the control structure can still be replicated in Promela, and this can be obtained directly

from a compiler's parse tree information. But this is still not enough, as in general codes the C statements manipulate data objects and this is not naturally done in Promela. To get around this, it is possible to embed C code directly in the verifier by letting the verifier know the code is "trusted". When a verification is performed using Spin, the verifier actually generates a C program from the specifications provided by Promela so there is no problem giving the compiler additional code that it can correctly interpret. This is of course fraught with possible difficulties, but at least one understands what assumptions have been made about the trusted code. Much of this process has been automated in software called MODEX (for model extractor) [8] that can guide a user in generating a model from generic code. Because this process assumes the embedded C code is trusted, there is the potential for all sorts of problems if in fact it contains bugs. One (somewhat labor intensive) solution to this is the use of embedded assertions in the Promela code that will allow execution of the trusted code provided some precondition is verified. This makes it possible, for example, to check for common bugs like dereferencing null pointers or allowing out of bound array operations that can lead to buffer overflow security flaws. Some of these checks could presumably be inserted in an automated fashion.

4.5 Relationship to Hyper-Properties

As a generalization of the type of approach described above we were briefed by Prof. Fred Schneider on the possible use of a new concept called hyper-properties as a way of describing security considerations. Typically in a given computing system a "security policy" is expressed in terms of confidentiality (you can't get or manipulate data that you're not entitled to get or manipulate), integrity (the data or

computation will not change in some non-deterministic way) and availability (the system responds in a timely way to commands). As noted by Schneider, however, it is difficult to have a precise definition of these. For example these concepts are not necessarily disjoint requirements, and they don't form a basis for expressing security policies in general. Finally, it is difficult to verify in the sense under discussion. An important contribution in this regard was made by Lamport [12] who expressed the notion of safety and liveness which were alluded to previously. Combined with the notion of linear temporal logic, as discussed briefly above, it became possible to make precise statements and prove them (using programs like Spin) for both safety and liveness propositions.

Schneider has generalized this notion of property verification to the more complex notion of policy verification. He notes that security policies are not properties but instead depend on sets of execution traces that originate from possibly different users of a system. For example, the idea of "non-interference" where commands of privileged users should have no effect on the observations of non-privileged users is not a safety property per se, but is instead a relation among execution traces and is really a requirement in a vein very similar to our earlier discussion about the issues with the Needham-Schroeder authentication protocol. Schneider calls such sets of traces hyper-properties and shows that we can express security policies in terms of such hyper-properties in a logically complete way.

This is useful in that it provides a common language for reasoning not only about properties of programs but executions of systems and enforcement of policies. This does not get around the essential complexity of verifying these properties or hyper-properties, and indeed, makes the search space bigger, but it can provide a foundation that could guide future developments, and this is urgently needed.

To conclude, the above discussion, while far from complete, does provide a connection between cyber-security and its underlying science, which in turn is connected to fundamental issues in model verification. There would seem to be no panacea for analysis of cyber-security but the foundations, and, to some extent, the methodologies for addressing specific questions do exist in the computer science community. What seems to be missing is a more direct connection to the day-to-day practice of programming. As more is learned about systematic examination of cyber-security, there is a need to translate the developments into tools that can be used by those writing software. In turn, this software must be continuously interrogated via techniques such as those described above. This will not cure the problem but it should improve the current state of practice.

```
active proctype I_intruder()
{
...
bit kNa, kNb;
bit k_Na_Nb__A ;          /* {Na, Nb}{PK(A)} */
bit k_Na_A__B ;           /* {Na, A}{PK(B)} */
bit k_Nb__B ;             /* {Nb}{PK(B)} */
mtype x1, x2, x3 ;

do
:: ca ! B, gD, A, B    :: ca ! B, gD, B, B
:: ca ! B, gD, I, B    :: ca ! B, A, A, B
:: ca ! B, A, B, B     :: ca ! B, A, I, B
:: ca ! B, B, A, B     :: ca ! B, B, B, B
:: ca ! B, B, I, B     :: ca ! B, I, A, B
:: ca ! B, I, B, B     :: ca ! B, I, I, B
:: ca ! (kNa -> A : R), Na, Na, A
:: ca ! (((kNa \&\& kNb) || k_Na_Nb__A) -> A : R), Na, Nb, A
:: ca ! (kNa -> A : R), Na, gD, A
:: ca ! (kNa -> A : R), Na, A, A
:: ca ! (kNa -> A : R), Na, B, A
:: ca ! (kNa -> A : R), Na, I, A
:: ca ! ((kNa || k_Na_A__B) -> B : R), Na, A, B
:: ca ! (kNa -> B : R), Na, B, B
:: ca ! (kNa -> B : R), Na, I, B
:: ca ! (kNb -> B : R), Nb, A, B
:: ca ! (kNb -> B : R), Nb, B, B
:: ca ! (kNb -> B : R), Nb, I, B
:: cb ! ((k_Nb__B || kNb) -> B : R), Nb, B
:: ca ? _, x1, x2, x3;
        if
        :: (x3 == I) -> k(x1); k(x2)
        :: else -> k3(x1,x2,x3)
        fi;
::  cb ? _, x1, x2;
        if
        :: (x2 == I) -> k(x1)
        :: else -> k2(x1,x2)
        fi;
od
}
```

Figure 4-7: Promela code for the intruder process

Received message	**Learned item**
$\{Na, A\}PK(I)$	Na
$\{Na, A\}PK(B)$	$\{Na, A\}PK(B)$
$\{Na\}PK(I)$	Na
$\{Nb\}PK(I)$	Nb
$\{gD\}PK(I)$	-
$\{A\}PK(I)$	-
$\{B\}PK(I)$	-
$\{I\}PK(I)$	-
$\{Na\}PK(B)$	$\{Na\}PK(B)$
$\{Nb\}PK(B)$	$\{Nb\}PK(B)$
$\{gD\}PK(B)$	$\{gD\}PK(B)$
$\{A\}PK(B)$	$\{A\}PK(B)$
$\{B\}PK(B)$	$\{B\}PK(B)$
$\{I\}PK(B)$	$\{I\}PK(B)$
$\{Na, Nb\}PK(I)$	$Na,\ Nb$
$\{Nb, Nb\}PK(I)$	Nb
$\{gD, Nb\}PK(I)$	Nb
$\{Na, Nb\}PK(A)$	$\{Na, Nb\}PK(A)$
$\{Nb, Nb\}PK(A)$	$\{Nb, Nb\}PK(A)$
$\{gD, Nb\}PK(A)$	$\{gD, Nb\}PK(A)$
$\{A, Nb\}PK(I)$	Nb
$\{B, Nb\}PK(I)$	Nb
$\{I, Nb\}PK(I)$	Nb
$\{A, Nb\}PK(A)$	$\{A, Nb\}PK(A)$
$\{B, Nb\}PK(A)$	$\{B, Nb\}PK(A)$
$\{I, Nb\}PK(A)$	$\{I, Nb\}PK(A)$

Figure 4-8: Table of items that the intruder can learn

Message	Needed knowledge (besides initial knowledge)
$\{Na, A\}PK(B)$	Na or $\{Na, A\}PK(B)$
$\{Na, B\}PK(B)$	Na or $\{Na, B\}PK(B)$
$\{Na, I\}PK(B)$	Na or $\{Na, I\}PK(B)$
$\{Nb, A\}PK(B)$	Nb or $\{Nb, A\}PK(B)$
$\{Nb, B\}PK(B)$	Nb or $\{Nb, B\}PK(B)$
$\{Nb, I\}PK(B)$	Nb or $\{Nb, I\}PK(B)$
$\{gD, A\}PK(B)$	-
$\{gD, B\}PK(B)$	-
$\{gD, I\}PK(B)$	-
$\{Na, A\}PK(A)$	Na or $\{Na, A\}PK(A)$
$\{Na, B\}PK(A)$	Na or $\{Na, B\}PK(A)$
$\{Na, I\}PK(A)$	Na or $\{Na, I\}PK(A)$
$\{A, A\}PK(B)$	-
$\{A, B\}PK(B)$	-
$\{A, I\}PK(B)$	-
$\{B, A\}PK(B)$	-
$\{B, B\}PK(B)$	-
$\{B, I\}PK(B)$	-
$\{I, A\}PK(B)$	-
$\{I, B\}PK(B)$	-
$\{I, I\}PK(B)$	-
$\{Nb\}PK(B)$	Nb or $\{Nb\}PK(B)$
$\{Na, Na\}PK(A)$	Na or $\{Na, Na\}PK(A)$
$\{Na, Nb\}PK(A)$	(Na and Nb) or $\{Na, Nb\}PK(A)$
$\{Na, gD\}PK(A)$	Na or $\{Na, gD\}PK(A)$

Figure 4-9: Table of items that the intruder needs to obtain

```
spin -a needham_orig.pml
cc -O2 -o pan pan.c
./pan -a
pan:1: claim violated! (at depth 30)
pan: wrote needham_orig.pml.trail

(Spin Version 5.2.5 -- 11 June 2010)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim             +
        assertion violations    + (if within scope of claim)
        acceptance   cycles     + (fairness disabled)
        invalid end states      - (disabled by never claim)

State-vector 60 byte, depth reached 31, errors: 1
      282 states, stored
      632 states, matched
      914 transitions (= stored+matched)
     1570 atomic steps
hash conflicts:         0 (resolved)

    2.501       memory usage (Mbyte)


pan: elapsed time 0.031 seconds
pan: rate 9096.7742 states/second
```

Figure 4-10: Output from a Spin verification analysis of the Needham-Schroeder protocol Promela specification

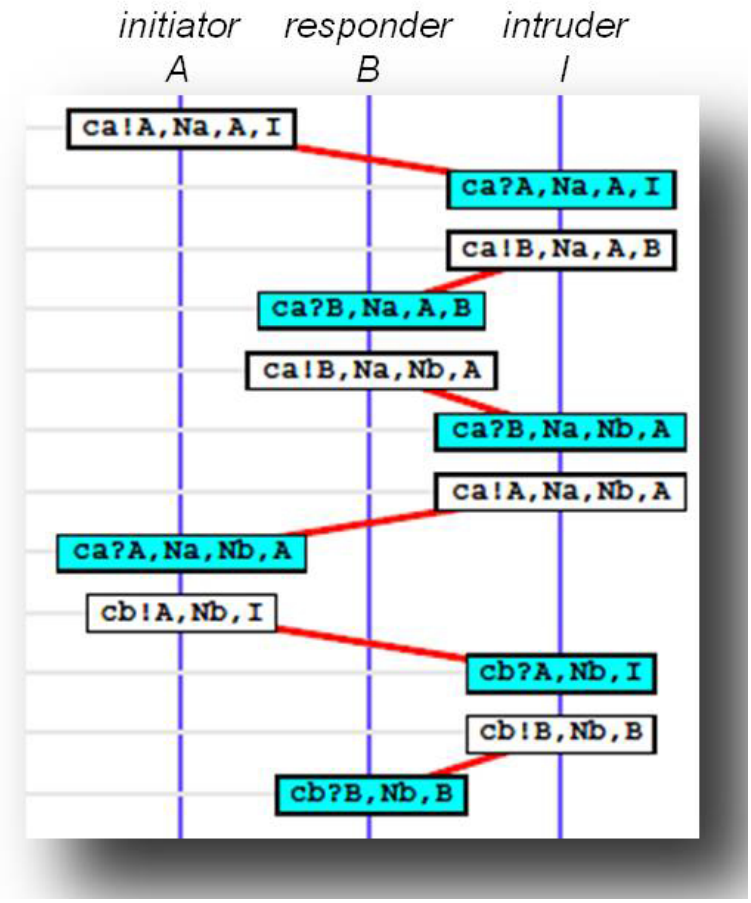

Figure 4-11: A diagram of the interaction leading to the invalidation of the Needham-Schroeder protocol

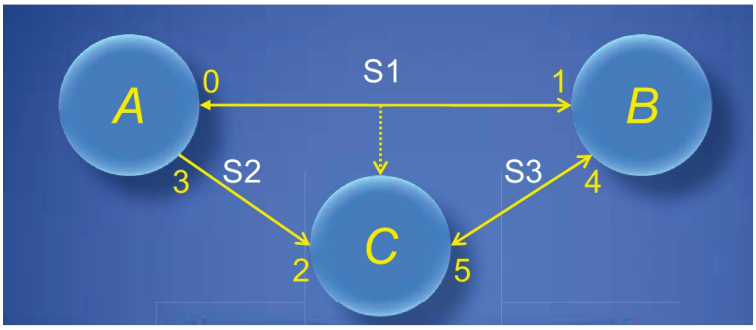

Figure 4-12: CCITT X.509 protocol exchange

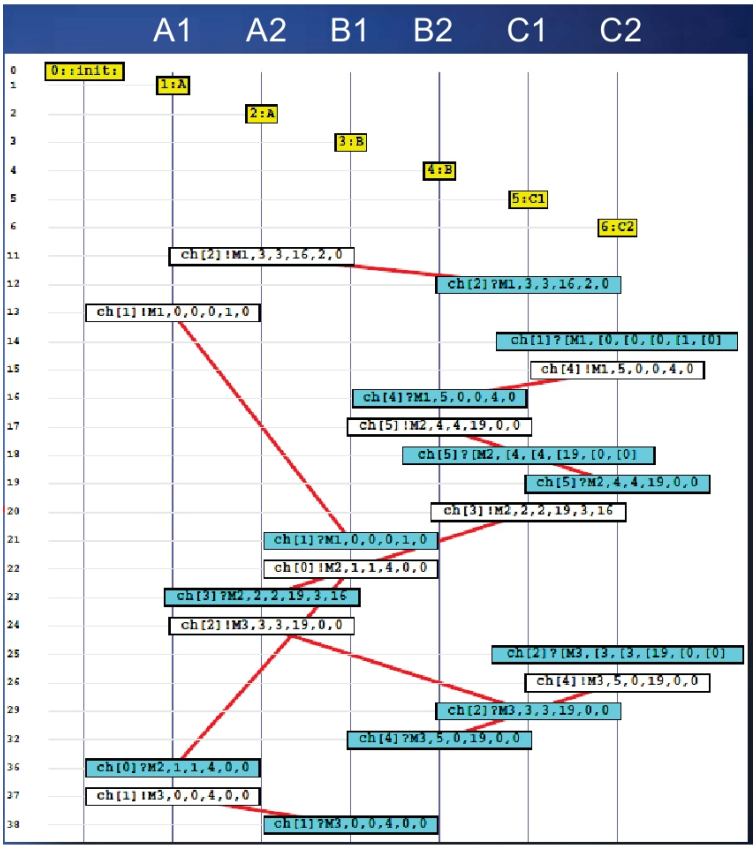

Figure 4-13: A Spin trace of a multi-session CCITT X.509 exchange showing violation of authentication

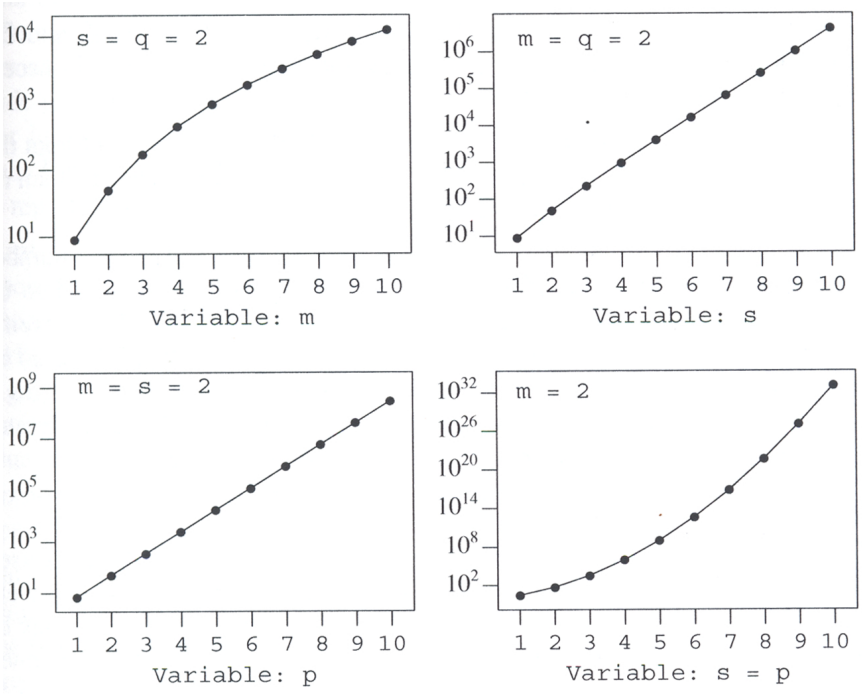

Figure 4-14: Scaling behavior for a collection of channels in Spin

```
#include <stdio.h>
 int main (void) {
   int lower, upper, step;
   float fahr, celsius;

   lower = 0;
   upper = 300;
   step = 20;

   fahr = lower;
   while (fahr <= upper) {
     celsius = (5.0/9.0) * (fahr -32.0);
     printf (... stuff...);
     fahr = fahr + step;
   }
}
```

Figure 4-15: C code for a simple Fahrenheit-Celsius converter

5 THE IMMUNE SYSTEM ANALOGY

It has long been recognized that the problem of protecting computers and computer networks from unwanted intrusions leading to takeovers is somewhat analogous to that of the human immune system protecting cells, tissues, organs and organisms from invading pathogens. We explore this analogy and the work in the cyber-security field that it has motivated. As we will see, the analogy is far from perfect, but there are important lessons that have been learned to date and additional concepts that may help guide future work.

5.1 Basic Biology

The study of the immune system is a vast scientific endeavor that has generated a wealth of information regarding the components of human immune response and the logical communication and computational network that coordinates their action. It is clearly not possible here to review immunology in general, so we will only explain pieces that seem to have some relevance to thinking about cyber-security. At the lowest level, immune response is launched against a predetermined set of targets. The components responsible for this response are usually denoted collectively as the innate immune system. The innate system is older in evolutionary terms and reacts to a fixed set of pathogens. At least for higher animals, this set repertoire response does not appear to be sufficient.

Layered above the innate system is the adaptive immune system. The major players here are lymphocytes in the form of B-cells and T-cells, each of which consists of several subtypes. These cells attempt to discover pathogens by recognizing ei-

ther their direct physical shape or their unique peptide subsequences. The most basic mechanism at work here is that of clonal selection. Cells are generated with a large variety of detectors, created by a sophisticated random genetic processing scheme. The initial B and T cells are naive and remain so until they find a match for their specific pattern detector (an epitope for the B-cell, a short peptide sequence for the T cell). Once the pattern is detected, the cell may become activated (depending also on other signals; see later), leading to large-scale proliferation so as to fight off the infection, or may be reserved as a memory cell to retain information about this particular pathogen for long-term future use. Activated cells have a relatively short lifetime and so the large-scale immune response is shut down rapidly once the pathogen is no longer detectable.

Unlike the innate system, the adaptive immune system does not operate with a fixed list of pathogenic patterns. The key issue is therefore how to prevent the misinterpretation of human cells and human peptides as being foreign. To do this, immature T-cells go through a testing period during which they attempt to bind to displayed self-peptides. Cells that bind too strongly are rejected, imprinting a negative self-image on the T-cell repertoire. This process is enabled by the fact that the set of self-peptides does not change in time; an allowed T-cell today will be allowed tomorrow and not cause auto-immune disease. As we discuss later, this is one of the places where the cyber problem may be more difficult.

From the above discussion and indeed from simple Biology 101 introductions to the immune system, one might get the impression that individual lymphocytes act as independent sentries, each guarding its own randomly determined gate. In fact, there is a great deal of cell-cell communication which gives rise to something more akin to an integrated defense network. For example, classes of T-cells

known as helper cells are involved in the activation of B-cells and the switching of some of these to become active secreters of antibodies (which directly bind epitopes and mark their carrier for destruction). It is not enough for a B-cell to detect pathogens, but instead there must be a coincident detection by T-cells. Incidentally, this is why it is enough to get rid of T-cells that bind self-peptides without similarly culling B-cells. This might be a mechanism for combining different types of information about a pathogen or simply a way of creating a sharper threshold for response (or both). Some of the coordinating cell-cell interactions take place by direct contact and others rely on the secretion and subsequent detection of a whole host of chemical communication agents (cytokines). There is even a complex logic to how antibodies are coordinated (the idiotypic antibody network). Exactly how the whole system operates is far from being understood; we are far from understanding the logic underlying many of these higher-level strategies.

Ultimately, the immune system guards against a variety of different types of pathogens, including viruses, bacteria and eukaryotic parasites. The enemy is always changing in the normal course of events, but at a rate determined by Darwinian evolution, not conscious planning. In the case of some agents, evolution will occur sufficiently slowly such that the immune memory will suffice to prevent them from gaining an initial foothold during the entire course of an individual's life. Of course, for some viruses such as influenza, the effective rate of variation is such that the memory is only of limited value. But even here, the new variants do not change in direct response to the defense mounted against their predecessors.

The goal of the immune system is of course to protect the whole organism. In its battle, individual cells are always expendable. Even in the brain (where, just

for the sake of accuracy, one should point out that the role of the immune system is taken up by the microglia), it is unlikely that individual cells are all that important; this is certainly true of all other organs. So, killer T-cells are tasked with the destruction of cells that show evidence of being taken over by a virus or intracellular parasite. Interestingly, individual cells become more indispensable as we go down the evolutionary ladder to invertebrates with their innate response. The ultimate limit here is C. Elegans where all cells have stereotyped roles that cannot be assumed by others. Strikingly, there are no known C. Elegans viruses; whether these facts are connected is not at all clear.

At an even more macroscopic level, not even organisms are sacrosanct. Populations often exhibit significant genetic variability so as to better survive attacks that happen to be reliant on specific biochemical details. One example concerns the fact that genetic variations of certain T-cell surface receptor proteins seem to render some individuals relatively immune to HIV infection. One can imagine a prehistoric version of the HIV story where the lack of medical knowledge led to the deaths of all individuals that did not by accident happen to be immune. The immune system itself takes advantage of variation in some of its important components; for instance, the MHC protein responsible for displaying peptide sequences to inquiring T-cell receptors can vary significantly from person to person,

5.2 Learning from the Analogy

What can we learn about design strategies and research protocols for cyber-security based on its analogy with immunology? This question has been addressed most forcefully by Forrest [7] and co-workers and, indeed, she was the one who briefed

us during our study. Our discussion below will focus on a number of different issues, with the overall perspective that this analogy can be quite useful for obtaining general guidance as long as one does not push it down to too fine-grained a level. At a fine-grained level, important technical differences make it likely that merely mimicking the biological system would not be a workable strategy.

5.2.1 The need for adaptive response

As mentioned earlier, higher animals cannot get by with a response predicated upon having a fixed list of bad actors. Unfortunately, that seems to be the approach taken by many commercial vendors of anti-virus software. These programs look for known virus code, suspect IP addresses, and other specific network infiltration signatures. These cannot possibly keep up with rapidly evolving threats and, indeed, some malware writers test their code to ensure that it is not detectable by these innate schemes.

At the most abstract level, studying the immune system suggests that cyber-security solutions will need to be adaptive, incorporating learning algorithms and flexible memory mechanisms. This argument leads us to expect that exact solutions to the problem based on precise definitions of vulnerability and liveness, will not easily translate into quantifiable security and verifiable code. Computer systems are too complex and too interactive for us to be able to avoid completely the "squishiness" so evident in living matter.

Adaptive solutions are expensive in terms of needed resources. Approximately 1% of human cells are lymphocytes, reflecting a rather large commitment to immune defense. One should therefore expect that significant amount of computational

power would be needed to run cyber-security for a typical network or cluster. However, deploying increased computational power could be used to advantage by DoD.

5.2.2 A mix of sensing modalities

The cyber-security literature seems to focus on using one sensing modality to detect and/or deter aberrant behavior. Thus, there are separate papers on network intrusion detectors based on traffic analysis, papers on using system call sequences to detect actions not intended by the legitimate user, papers on using diversification of low-level implementations of high-level code etc. The immune system teaches us that it might be important to use all of these strategies simultaneously and synergistically. Pathogens can give rise to either intracellular or extra-cellular signals (or both) and both are searched for by the immune system using both innate and adaptive approaches and utilizing a whole range of specialized hardware. One might always hope for a cyber magic bullet but the immune systems functions quite well even though it has yet to find a single fool proof strategy. Learning is necessary, but need not be restricted to bio-mimicry. At the heart of any adaptive system is a learning algorithm that allows for the discrimination of allowed actions and disallowed ones, even though these cannot be specified a priori in the system design. As already mentioned, the immune system uses a strategy based on negative selection. This very notion has been suggested for a cyber system which acts as an "information immune system" applied to bytes in memory. This is illustrated schematically in Figure 5-1; the circles represent detectors, generated around the "x" characters. Some of these detectors are not generated since they are ruled out by being inside the blue allowed region.

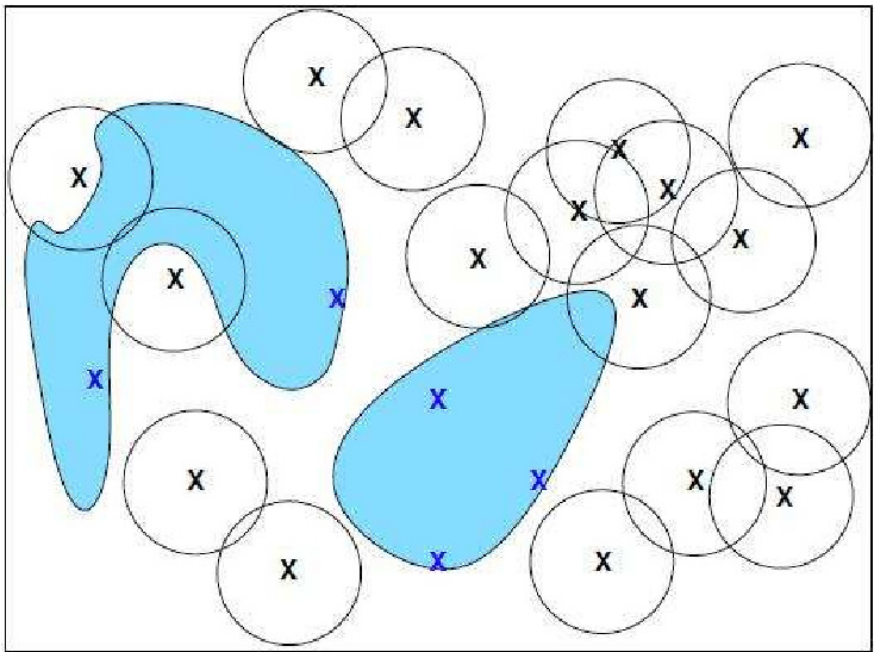

Figure 5-1: The immune system cartoon

In general, we do not see why the specifics of the immune learning system should be taken over to the cyber problem. There are many machine learning paradigms and some of these may be more suitable to the problem at hand; this needs to be studied by a systematic experimental program. There is indeed a significant amount of work in this area, testing one specific algorithm versus some specific dataset, but somehow these case studies do not seem to be leading to any general insights or conceptual framework for understanding what works when and why.

5.2.3 The need for controlled experiments

One way to help the field consolidate lessons learned from individual experiments is for these different studies to be conducted under identical circumstances with fixed datasets. This is a difficult challenge, as seen in the following statement by

Forrest:

> "The complexity of modern computing environments poses a serious challenge to replicating results and comparing multiple experiments. It is surprisingly difficult to document precisely the conditions under which an experiment is run and seemingly trivial differences in system configuration can affect the outcome dramatically."

Other reasons as to why there has been little progress towards an experimental discipline concerns the fact that the threat evolves rapidly and computer systems change rapidly as well and that, therefore, testing a security system that runs on an old platform against last year's malware is not considered informative. If there is ever going to be a scientific basis for security choices, this need for proceeding to the new before understanding the old has to be overcome. More generally, modern biology, including immunology, faces a similar challenge in designing experiments with protocols that are tight enough to allow for reproducibility. Some of the ideas that have worked include doing related in vitro experiments, experiments in model organisms (such as mouse) and then correlating the results to less well-controlled data on the problem of real interest (human medicine). Translating these notions to cyber-security, this would mean having a mixture of small-scale experiments on controllable platforms with shared datasets, having perhaps some dedicated testbeds, which could allow larger scale results but always with the same system configuration. Claims that experiments on testbeds would be "contrived" and would not directly translate to useful engineering solutions are true in an absolutist sense, but the goal is to advance conceptual understanding, not build a marketable solution. Finally, results here would be correlated to behavior of real systems, in the same manner as modern medicine does when it compares clinical

data to biology experiments.

5.2.4 Time scale differences

One important issue for proceeding with solutions based on the immune system analogy concerns the differences in some time scales between the two problems. On the "self" side, usage of a computer system may change significantly over its lifetime and of course networks can change in size and connectivity patterns. This means that the learned determination of what is allowed must be allowed to change too. One will need to find an approach that allows changes on a slow time scale, but still can detect aberrant behavior, which hopefully looks like a sudden alteration. The only immune system analogy to this additional difficulty is that humans have to learn to be relatively insensitive to commensal microorganisms that reside in our gut and act symbiotically; this learned insensitivity has to be done initially at birth and then allowed to drift over a lifetime, as the exact internal biome changes over time. How exactly this is accomplished is not clear.

On the opposite side of the battle, malware composers can be extremely rapid in adapting to a new defensive strategy and do so with purposefulness not matched by evolutionary attempts to evade the immune system. This means, for example, that it makes little sense to openly reveal the exact inner workings of a fielded security system. It is basically an experimental question, to be investigated with red team methods, to see which detection schemes and which learning algorithms are most insensitive to active anti-defense measures.

5.2.5 Responses to detection

In addition to detecting pathogens, the immune system acts to contain the infection. It does this in a thresholded manner, using the cell-cell communication system to make 'intelligent' decisions regarding when to proliferate specific attack agents, when to declare an alert (that is, inflammation) and when to return to normal. Part of the strategy is based on the fact that killing individual cells is not very costly for overall organism function as long as the system is sure that the cell really exhibits foreign agent takeover. In the interest of avoiding autoimmune problems, the system is set up to have a very low false positive rate at the expense of allowing small-scale infections to remain undetected.

We are not quite at the era when individual computers in a network are completely expendable; this is probably more true of individual processors in a cluster. Perhaps as we move toward a cloud computing paradigm, this situation will change and the proper response to detection will be to just turn off the offending computer until it can be cleaned up. To the extent that we follow the analogy and take drastic action but only after a threshold is passed, one can imagine co-evolution of malware so as to be more benign. In other words, the final state of a cyber-security effort might be that takeover still occurs, but the measurable effects of infection are small while the computer is still performing useful tasks for the "botmaster". The extent to which individuals will be willing to have their personal systems disallowed from the network if they are being used only rarely to send spam, is an interesting sociological question beyond the scope of what is considered here.

5.2.6 Final points

The idea that diversity is a good defense for some attacks has already been implemented in the cyber-security sphere. The extent to which this is a successful strategy depends on the goals. If we are trying to protect a majority of computers from certain classes of malware, this apparently can be accomplished; if we are trying to rid the web of spam sent out by botnets, the fact that, say, 1% of computers can still be attacked means that we have failed.

Finally, the immune system has help, as we have managed to make hygiene part of our social contract. As is often quipped, plumbers made a larger contribution to public health over the last century than doctors. We shun people who do not wash their hands after being in the toilet, who sneeze in public without covering their mouth, who do not voluntarily stay home from school or work when suffering from the flu. Perhaps the final lesson to be learned is that we should move towards societal frowning upon bad security practices such as keeping a computer on the network even as is it being used by spammers, picking easily breakable passwords, downloading programs from unfamiliar websites. In some sense we attempt to do this today, but, in the case of medicine, the adversary typically does not mutate in response into something more malevolent in response to our attempts to exercise hygiene.

6 CONCLUSIONS AND RECOMMENDATIONS

There is a science of cyber-security. Because it is a science with adversaries, it uses, and will use, many different tools and methods. For the future, as far as can be discerned, there will be new attacks on old technologies, and new technologies that need to be defended.

The science seems underdeveloped in reporting experimental results, and consequently in the ability to use them. The research community does not seem to have developed a generally accepted way of reporting empirical studies so that people could reproduce the work and use the results.

There have been lots of reports on the need for R&D for cyber-security. A good list is at `http://www.cyber.st.dhs.gov/documents.html`. There is universal agreement that more work is needed. We couldn't find anyone who even felt that there is already enough known, and all that is needed is to apply current knowledge. Cyber-security is a problem that may be manageable, but not only is it not solvable, there's no agreement that it is being managed well.

We endorse the proposal given to our sponsor in an IDA report titled "Cyber-Security Technology Initiatives," to "establish multiple cyber-security science base(d) centers and projects within universities and other research centers. These programs should have a long time horizon and periodic reviews of accomplishments."

Centers and programs sponsored by the Department of Defense have several attractive features. First, they give the sponsors access to the best ideas and people. Informal relationships are particularly important in this area, where both the state

of knowledge and the facts on the ground are changing rapidly. Second, it gives the sponsor the chance to bias the work towards their versions of common problems. Third, there is an opportunity for these centers and programs to leverage a unique collection of resources internal to the DoD. Among these are the defensive data and experience from running internal networks, presumably now under the aegis of Cyber Command. This is likely synergistic with the Information Assurance program at the NSA. Also DARPA has specific shorter-term projects. The difference between the proposed centers and DARPAs projects is that the centers would be expected to make steady progress on a broad set of topics, rather than limit themselves to revolutionary ideas or to try to solve the latest cyber-security crisis.

In addition the centers could act as connecting points for the software industry so as to accelerate the translation of new ideas (or even old ideas) into useful tools developers can apply automatically. We believe there is also a need for a concerted program to bring the results of the research community into wider use. History is not encouraging here. There are some very sophisticated approaches (model checking, type checking etc. as discussed previously) that can be used to assess and reason about the security of current systems, but they are not widely available today in the form of developer tools. There may be an insufficient market for private development of such tools and this may argue for a more activist role on the part of DOD in supporting future development.

DOD posed a number of questions as part of the study. We quote below in summary these questions and our responses.

1. *What elements of scientific theory, experimentation, and/or practice should*

the cyber-security research community adopt to make significant progress in the field? How will this benefit the community? Are there philosophical underpinnings of science that the cyber-security research community should adopt?

The most important attributes would be the construction of a common language and a set of basic concepts about which the security community can develop a shared understanding. Since cyber-security is science in the presence of adversaries, these objects will change over time, but a common language and agreed-upon experimental protocols will facilitate the testing of hypotheses and validation of concepts. The community can benefit here if there emerges a consensus on progress as well as more concrete notions of what future directions are most promising. At the same time, there must be a connection with practice "in the wild" (in the same sense as one progresses from animal model tests to possible clinical trials in medical research).

2. *Are there "laws of nature" in cyber space that can form the basis of scientific inquiry in the field of cyber-security? Are there mathematical abstractions or theoretical constructs that should be considered?*

   There are no intrinsic "laws of nature" for cyber-security as there are, for example, in physics, chemistry or biology. Cyber-security is essentially an applied science that is informed by the mathematical constructs of computer science such as theory of automata, complexity, and mathematical logic.

3. *Are there metrics that can be used to measure with repeatable results the cyber-security status of a system, of a network, of a mission? Can measurement theory or practice be expanded to improve our ability to quantify cyber-security?*

There are many metrics that could be productively employed, such as those that inform modern intrusion detection systems. But it must be understood that any such metrics are empirically based and statistical in nature; they cannot apply to scenarios that are not well defined. In particular, things that are not observed such as new attack approaches are not going to contribute to metrics. It is not possible to definitively measure a level of security as it applies to general operation of information systems. The repeatability of results hinges upon the imposition of community protocol standards and adherence to these criteria. At present, however, repeatability is not a central criterion for publication of results in cyber-security.

4. *How should a scientific basis for cyber-security research be organized? Are the traditional domains of experimental and theoretical inquiry valid in cyber-security? Are there analytic and methodological approaches that can help? What are they?*

   There is every reason to believe that the traditional domains of experimental and theoretical inquiry apply to the study of cyber-security. The highest priority should be assigned to establishing research protocols to enable reproducible experiments. These should provide clear descriptions of the initial conditions, the types of allowable threats and clear meanings for the security goals.

5. *Are there traditional scientific domains and methods such as complexity theory, physics, theory of dynamical systems, network topology, formal methods, mathematics, social sciences etc. that can contribute to a science of cyber-security?*

   There are several areas that are traditionally a part of computer science that

have contributed in the past to a deeper understanding of cyber-security and where increased future emphasis could bring continued improvement. The field of model checking seems particularly relevant in that one creates a model for the security of a given system or key kernel and then tests the assumptions using a well defined set of possible inputs. While the input space is potentially infinite, the benefit is that specific threats can be modeled. The area of cryptography has traditionally focused on provable aspects of secure communication with careful attention paid to the nature of assumptions. The application of code obfuscation and type theory also provide important insights into the construction of secure code. Finally, there is an important role for game theoretic approaches to security evaluation. For DOD, there is also value in exploiting secrecy as a factor in cyber defense. Examples include use of obfuscation as described above, development of DOD-specific security products and in general collecting and securing data about cyber attack that are not publicly shared.

6. *How can modeling and simulation methods contribute to a science of cyber-security?*

   There are a number of ways in which modeling and simulation can contribute. One is by using existing computational capability to continually test security assumptions on running systems. Another is to use concepts of virtual machines, etc. to provide well-defined test beds to explore in a controlled way the behavior of computational and security systems in the presence of well-defined attacks.

7. *Repeatable cyber experiments are possible in small closed and controlled conditions but can they be scaled up to produce repeatable results on the entire Internet? To the subset of the Internet that support DoD and the IC?*

   It is premature to ask this question, as there would appear to be little organization of the results of previous small scale experiments. As many of these were not performed with the goal of repeatability, it is important to first assess the utility of these smaller scale test beds before contemplating the scale-up to controlled wide area networks or to the Internet in general.

8. *What steps are recommended to develop and nurture scientific inquiry into forming a science of cyber-security field? What is needed to establish the cyber-security science community?*

   The establishment of interdisciplinary centers that connect academia, industry, national laboratories and the DOD would be an important step in this direction. Such centers should focus on cyber-security issues of particular relevance to DOD needs, but would also leverage the activities of other important contributing agencies such as DARPA and the Information Assurance efforts within the NSA. In all such enterprises it will be important to establish protocols developed for, and peer-reviewed by, the community so as to facilitate the communication and archiving of results.

9. *Is there reason to believe the above goals are, in principle, not achievable and if so, why not?*

   There is every reason to expect that significant progress can be made toward the above goals. One must first understand the nature of the scientific enterprise for cyber-security and characterize the objects under discussion. There is a great deal of valuable science that can be accomplished if an accepted

approach to discourse can be developed. While these primarily technical activities will not in and of themselves "solve" the cyber-security problem given that it has both technical and social aspects, they will significantly aid progress.

A APPENDIX: Briefers

Briefer	Affiliation	Briefing title
Steven King	DOD OSD	Cyber-Security: Is Science Possible?
Fred Schneider	Cornell University	Towards a Science of Security
John McLean	Naval Research Laboratory	The Science of Security - A Formal Perspective
Stephanie Forrest	Univ. New Mexico	Bridging CS and Biology for Cyber-Security
Grant Wagner	NSA	NSA Perspective: Application of SOS
Bob Meushaw	NSA	NSA Perspectives on the Science of Security
Kamal Jabbour	AFOSR	The Current Threat
Ron Rivest	MIT	Science of Security: Perspectives from Cryptography
Drew Dean	DARPA	
John Mitchell	Stanford Univ.	Security Modeling and Analysis
Jeff Shamma	Georgia Tech	Science of Security and Game Theory
Carl Landwehr	NSF	Science of Security: From Engineering to Science
John Manferdelli	Microsoft Research	Science of Security: Practice in the Information Marketplace
Milo Martin	Univ. of Pennsylvania	Science of Security: A Hardware Persepctive
Gerard Holzmann	Jet Propulsion lab	Protocol Analysis and Model Checking
John Chuang	UC Berkeley	Economics of Information Security
Roy Maxion	Carnegie Mellon Univ.	Experimentation in Cyber Security
Stefan Savage	UC San Diego	Computer Security: Science, Engineering, Economics, and a Bit of Reality
Peter Galison	Harvard Univ.	Augustinean and Manichaean Science

References

[1] D. Akhawe, A. Barth, P Lam, and others, Towards a formal foundation of web security. *Proc. 23rd IEEE Computer Security Foundations Symposium*, 2010.

[2] T. Alpcan and Başar T. , *Network Security: A Decision and Game Theoretic Approach*. Cambridge University Press, 2011.

[3] Al Bessey, Ken Block, et al., A few billion lines of code later: Using static analysis to find bugs in the real world. how coverity built a bug-finding tool, and a business, around the unlimited supply of bugs in software systems. *Communications of the ACM*, 53, 2010.

[4] Leventa Buttyan and Jean-Pierre Hubaux, *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2007.

[5] Michael R. Clarkson and Fred B. Schneider, Hyperproperties. In *CSF 2008: 21ST IEEE COMPUTER SECURITY FOUNDATIONS SYMPOSIUM, PROCEEDINGS*, pages 51–65. IEEE COMPUTER SOC, 2008. 21st IEEE-Computer-Securituy-Foundation Conference, Pittsburgh, PA, JUN 23-25, 2008.

[6] Ulfar Erlingsson, *Low-level software security: Attacks and defenses*. Springer Verlag, 2007.

[7] Stephanie Forrest, Binding cs and biology for cybersecurity, presentation to JASON.

[8] G. Holzmann and M. Smith, Extracting verification models from source code. In *Formal methods for protocol Engineering and distributed systems*, pages 481–497, 1999.

[9] Gerard Holzmann, Protocol analysis and model checking, presentation to JASON.

[10] Gerard J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2003.

[11] Audun Josang, Security protocol using spin. *Proc. TACAS96, LNCS*, 1996.

[12] L. Lamport, Proving correctness of multi-process programs. *IEEE transactions on software engineering*, 3(2):125–143, 1977.

[13] G. Lowe, Breaking and fixing the needham-schroeder public-key protocol using fdr. *Proc. TACAS96, LNCS*, 1996.

[14] P. Maggi and R. Sisto, Using spin to verify security properties of cryptographic protocols. *Proc. 9'th Spin Workshop, LNCS*, 2002.

[15] Yu. I. Manin, *A course in mathematical logic*. Springer, 1977.

[16] R. M. Needham and M.D. Schroeder, Using encryption for authentication in large networks of computers. *Commun. ACM*, 21, 1978.

[17] Ron Rivest, Presentation to JASON.

[18] Donald E. Stokes. *Pasteur's Quadrant*. Brookings Institution, 1997.

[19] Alan Gibbard and Hal R. Varian, "What Use Are Economic Models", Journal of Philosophy, 1978.

[20] Wikipedia. `wikipedia.org/wiki/Address\_space\_layout\_randomization`.