What Makes a Cryptographic Protocol Secure? The Evolution of Requirements Specification in Formal Cryptographic Protocol Analysis

Catherine Meadows

Naval Research Laboratory
Center for High Assurance Computer Systems
Washington, DC 20375
meadows@itd.nrl.navy.mil

Abstract. Much attention has been paid to the design of languages for the specification of cryptographic protocols. However, the ability to specify their desired behavior correctly is also important; indeed many perceived protocol flaws arise out of a misunderstanding of the protocol's requirements. In this talk we give a brief survey of the history of requirements specification in formal analysis of cryptographic protocols. We outline the main approaches and describe some of the open issues.

1 Introduction

It has often been pointed out, that, although it is difficult to get cryptographic protocols right, what is really difficult is not the design of the protocol itself, but of the requirements. Many problems with security protocols arise, not because the protocol as designed did not satisfy its requirements, but because the requirements were not well understood in the first place.

Not surprisingly, the realization of this fact has lead to a considerable amount of research in security requirements for cryptographic protocols. However, most of this literature is scattered, and unlike the topic of cryptographic protocol analysis in general, there is little existing survey work providing roadmaps to readers interested in learning more about the topic. In this paper we attempt to remedy this deficiency by providing a brief history and survey of the work that has been done in this area, and outlining what we consider to be some of the open problems.

Any scheme for expressing requirements should satisfy three properties:

1. It should be expressive enough to specify properties of interest.
2. It should be unambiguous, and preferably compatible with with some system for formal analysis.
3. It should be easy to read and write.

It will helpful to keep these three properties in mind as we proceed through our survey.

The paper is organized as follows. We begin in the next section by describing some of the early approaches to specifying cryptographic protocol requirements, including that of Burrows, Abadi, and Needham. In the third section, we describe some of the main current approaches to requirements in terms of a spectrum from extensional to intensional requirements. In the fourth section and fifth sections, we discuss two emerging areas of research: graphical languages for specifying cryptographc protocol requirements, and the expression of quantitative requirements. In the final section, we sum up what we believe to be some of the open problems, and conclude the paper.

2 Early Work in Cryptographic Protocol Requirements

Most of the existing approaches to applying formal methods to cryptographic protocol analysis stem ultimately from that of Dolev and Yao [9], who developed for the first formalization of the intruder model that is commonly used today. However, since Dolev and Yao's work and its immediate successors was mainly focussed on theoretical results about the complexity of cryptographic protocol analysis, only one type of requirement was considered, and that was the simplest: that some term or set of terms designated as secret should not be learned by the intruder. Some of the earlier work on automated cryptographic protocol analysis, such as the first versions of the Interrogator [24], also restricted itself to this limited definition of secrecy. Others, such as the earlier versions of the NRL Protocol Analyzer[20], allowed the user to specify security in terms of the unreachability of insecure states, in which it was possible to specify such a state in terms of words known by the intruder and the values of local state variables of the principles. However, the user was not given any further assistance in constructing requirments.

Probably the first formal cryptographic protocol analysis system to provide a real mechanism for constructing formal requirments was the belief logic of Burrows, Abadi, and Needham [5].

BAN logic does not address secrecy at all. Rather it confines itself to questions of authentication. Questions that BAN logic can be used to decide have to do with beliefs the participating principals could derive about origin and use of information such as:

1. Where does the information come from?
2. What is the information intended for?
3. Is the information new, or is it a replay?
4. Who else has these beliefs about the information?

One uses BAN logic by attempting to see which of these beliefs can be derived from an idealization of the protocol. The BAN logic does not dictate which beliefs a protocol should be able to satisfy; rather it is up to the protocol analyst to decide what beliefs a protocol should guarantee, and to determine it those beliefs can be derived from the protocol. Thus, one might require that Alice believe that

K is a good key for communicating for Bob, and that Bob believe that K is a good key for communicating with Alice, but one might or might not want to require that Alice believe that Bob believes that K is a good key for communicating with Alice, and vice versa. Thus BAN logic provides what it probably the first formal system for specifying cryptographic protocol requirements.

3 Safety Requirements for Cryptographic Protocols: Secrecy and Correspondence

In the early to mid-90's the approach to cryptographic protocol verification tended towards the application of general-purpose tools such as model-checkers and theorem provers. With this came the need to develop means for specifying the properties one was attempting to prove. Since, in general, researchers were now reasoning directly about messages passed in a protocol, rather than about beliefs that were developed as a result of receiving those messages, it now made sense to develop requirements in terms of messages sent and received rather than beliefs derived.

As is the case for requirements in general, requirements for cryptographic protocols tend to fall into two categories, extensional and intensional. Extensional systems provide a small set of generic requirements that can be defined independently of the details of any particular protocol. Intensional systems provide languages and techniques that can be used to specify requirements for specific protocols in terms of the protocols themselves. This concept was first discussed in detail in the context of cryptographic protocols by Roscoe in [27]. He noted that the earlier work in cryptographic protocol requirements, such as BAN, leaned to the extensional side, and he showed how one might specify intensional protocol requirements in CSP.

Requirements for cryptographic protocols also fall into two classes that are related to the properties that such protocols are intended to enforce: secrecy and correspondence. Secrecy requirements describe who should have access to data. Correspondence requirements describe dependencies between events that occur in a protocol, and are usually used to express authentication properties. These two types of requirements later turned out to be more closely related than one might think (both Syverson and Meadows [32] and Schneider [28] define secrecy requirements as a type of correspondence requirement), but for the moment we shall treat them as separate.

Of course, not all requirements can be characterized in terms of secrecy and correspondence. In particular, they are both safety properties, so any non-safety requirements (such as fairness and its relatives, which are relevant for many electronic commerce protocols) will not fall into either of these two categories. However, secrecy and correspondence cover most requirements relevant to authentication and key exchange, and thus make a good starting point.

At first, correspondence requirements appeared to be the most subtle and complex; thus the earlier work tended to concentrate on these. Moreover, the

emphasis was on extensional requirements and the ability to characterize a general notion of correspondence in a single definition. Probably the first work in this area was that of Bird et al [4]. In the introduction to their paper, they describe an error-free history of a protocol runs between two prinicpals A and B to be one in which all executions viewed by both parties match exactly one-to-one. This is idea is refined by Diffie, van Oorschot and Wiener in [8] to the idea of *matching protocol runs*, which says that at the time Alice completes a protocol the other party's record of the run matches Alice's. This notion was further refined and formalized by Bellare and Rogaway in [3] to the notion of *matching conversations*, which developed the idea in terms of a complexity-theoretic framework.

Such general of notions of correspondence can be very useful, but they do have a drawback. They can be used to determine whether or not information was distributed correctly, but they can not be used to determine whether or not all information that should have been authenticated was included in the run.

To see what we mean, we consider the attack found by Lowe [18] on the Station-to-Station protocol of [8]. The protocol is defined as follows:

1. $A \rightarrow B \ : \ x^{N_A}$
2. $B \rightarrow A \ : \ x^{N_B}, E_K(S_B(x^{N_A}, x^{N_B}))$
   where K is the Diffie-Hellman key generated by A and B.
3. $A \rightarrow B \ : \ , E_K(S_A(x^{N_B}, x^{N_A}))$

Lowe's attack runs as follows:

1. $A \rightarrow B \ : \ x^{N_A}$
   An intruder I intercepts this message and forwards it to B, as if it came from C.
2. $B \rightarrow I_C \ : \ x^{N_B}, E_K(S_B(x^{N_B}, x^{N_B}))$
   The intruder forwards this message to A.

Thus, at the end of A's run, A believes that it shares a key with B. B, however, thinks that C is trying to establish a connection with it, and it will reject A's final message when it receives it, because it is expecting confirmation from C, not A. On the other hand, the protocol does satisfy the matching protocol runs definition of security, since A's picture of the authenticated portions of the messages is the same as B's. Indeed, this is the protocol used to illustrate the concept by Diffie, van Oorschot, and Wiener in [8].

Lowe's attack, of course, does not mean the Station-to-Station protocol is insecure. (Indeed, this very feature of that protocol is seen as a desirable property in the latest version of IKEv2, the proposed replacement to the Internet Key Exchange protocol [17]). All it does is show that, if the name of the intended recipient is not included in the responder's message, a definition of security that is specified in terms of conditions on correspondence between messages will not catch lack of agreement on information that is never sent.

Lowe's solution to this problem in [18] was to strengthen the matching protocol runs requirement to include the condition that when A completes a protocol

run with B, then not only should the two protocol runs match, but B should believe that he has been running the protocol with A. In a later paper, [19], he developed this idea further, developing a hierarchy of authentication requirements which gave conditions of varying degrees of strictness on the conclusions a principal A could draw about B's view of the protocol after completing the protocol with B. These were then formalized using the process algebra CSP.

The least restrictive requirement Lowe gave was liveness, which simply requires that, when A completes a run of the protocol, apparently with B, then B has also been running the protocol. Moving further up the hierarchy, we require A and B to agree on messages sent as well as identities (this requirement correspondes roughly to matching protocol runs), to agree on the roles they are playing, to agree on the values of specific data items, and so forth.

We see now that we are moving away from extensional requirements that can be specified independently of the protocol, and more to intensional requirements. If principals need to agree on specific data items, we need to specify what these data items are, and where they occur in the protocol. The next step would be to specify the conditions on events that occur in protocols. Indeed, it should be possible to specify the types of requirements we are interested in using the temporal logics that are generally used to provide correctness specifications for model checkers.

This is the sort of reasoning that lay behind Syverson and Meadows' development of a requirements language for the NRL Protocol Analyzer [32], which eventually became known as the NRL Protocol Analyzer Temporal Requirements Language (NPATRL). The idea is to develop a simple temporal language that can be used to specify the type of requirements that are commonly used in authentication and key distribution protocols. The atomic components of the language correspond to events in the protocol (e.g. the sending and receiving of messages, or the intruder's learning a term). Besides the usual logical connectives, it contains only one temporal operator, $\diamondsuit$, or "happened previously." The use of this single logical operator reflects the fact that most correspondence requirements can be expressed in terms of events that must have or must have not occurred before some other events.

Although NPATRL is a very simple language, we have found it useful for specifying some widely varying types of cryptographic protocols. These include key distribution and key agreement protocols [30, 31], complex electronic commerce protocols such as SET [22], and, most recently, group key distribution protocols [23].

One interesting result of our experience is that we have found NPATRL increasingly useful for specifying complex secrecy requirements as well as complex authentication requirements. Early requirements for secrecy simply designated some information, such as keys, as secret, and all that needed to be guaranteed was that these keys would not be available to an intruder. However, more recently, requirements such as perfect forward secrecy put other conditions on an intruder learning a term. Perfect forward secrecy requires that, if a master key is compromised, then an intruder can only learn a session key after the

compromise, not before. Such a requirement is straightfoward to specify using a temporal language.

Of course, temporal logics are not necessary in order to specify these types of requirements. Other formalisms will work as well. For example, Schneider [28] defines authentication in terms of the messages that must precede a given message, and secrecy in terms of another correspondence requirement, that the intruder should not learn data unless that data was explicitly sent to the intruder. Both of these are formalized in CSP.

Another approach to requirements, taken by Focardi et al. [12], allows one to specify requirements of varying degree of generality. They make use of notions derived from noninterference. Their notion of correctness, Generalized Nondeducibility on Composition, or $GNDC$, is defined as follows.

We let P be a process representing a cryptographic protocol operating in the absence of an intruder. Let $(P||X)$ denote the composition of P with an intruder X. Let α denote a function from processes to processes where $\alpha(P)$ is a process describing the "correct" behavior of P. Let $\approx$ denote a preorder. Let C denote the set of channels between honest principals, and let $Q\ C$ denote the restriction of a process Q to C. Then a process satisfies $GNDC^{\alpha}_{\approx}$, if, for all intruders X

$(P||X)\backslash C \approx \alpha(P)$

In the case of that α is the identity function and $\sim$ is trace equivalence, the property becomes NDC, or Nondeducibility on Composition, which requires that the traces produced by the process in composition with an intruder be the same as the traces produced by the process in the absence of the intruder. This can be thought of as an information-flow property in which the intruder and P play the apart of High and Low, respectively, corresponding to the standard multilevel application of noninterference for multilevel security [14]. NDC, since it requires that a process behave in the presence of an intruder exactly at it would behave in the absence, is more stringent than any of the other requirements that have been discussed in this section. As a matter of fact, we can consider it the most stringent definition possible, closely akin to the fail-stop definition of protocol security of Gong and Syverson[13]. Moreover, $GNDC$ provides a framework that allows one to specify less restrictive requirements such as the various forms of correspondence discussed earlier, and the types of requirements that would be defined in a temporal language such as NPATRL. Thus $GNDC$ can be thought of as providing a general framework for requirements, including requirements that go beyond the usual notions of correspondence, such as liveness.

Another technique that deserves mention is the notion of using type theory to specify security requirements and evaluate the correctness of protocols [1, 15]. Here components making up a protocol, such as data, channels, etc. are assigned different types, such as secret or public. Rules are also developed for deriving types from the results of applying operations, such as encryption, on data. Security violations can be defined in terms of typing violations, such as a piece of data type public appearing on a channel of type public. Most of this work has been applied to the type-checking of secrecy properties, but Gordon and Jeffrey [15, 16] have developed ways of applying it to correspondence properties,

specifically one-to-one (each event of a certain type should be preceded by one and only one event of a certain other type) and one-to-many (each event of a certain type should be preceded by at least one event of a certain type). Since the types are supplied as part of the protocol specification, this application of type theory gives a nice way of incorporating a requirements specification as an annotation on the protocol.

4 Graphical Requirements Languages

Languages and frameworks such as NPATRL and $GNDC$ allow us increasing flexibility and expressiveness for specifying requirements. But, the ability to specify more complex and subtle requirements also has a cost; the requirements become more difficult to comprehend and write. In this section we discuss two graphical approaches to increasing the ease of handling such specifications that make use of some of the common features of cryptographic protocols and their requirements.

The first of these is known as *Strand Space Pictures* [10]. Strand spaces [11] are a well-known and popular model for cryptographic protocol analysis, in which the actions of principals are modeled in terms of graphs. A *strand* represents a principal executing a role in a protocol. The sending and receiving of messages is represented by positive and negative nodes. Nodes that represent one event immediately preceding another on a strand are connected by double arrows. A *bundle* is a collection of strands, in which positive send nodes can be connected to negative receive nodes via a single arrow if the message sent matches the message received. This model facilitates the graphical representation of protocols, and [10] actually describes a number of ways in which the graphical features of strand spaces could be used, but the one of most interest to us is the way in which they can be used to represent requirements. Using strand space representation of protocols, it is possible to represent correspondence requirements in terms of relative placement of strands. Thus, if we want to specify a correspondence requirement which requires that if certain messages are accepted, then other messages were sent previously, we can represent sending and receipt of the messages we are interested in by portions of strands, and we can use the placement of the strands (so that earlier nodes appear above later ones) to indicate which events we want to occur before others.

The strand space pictures methodology, was never, as far as we know, developed into a full-fledged procedure with well-defined ways for representing major classes of requirements. However, in [10] the authors give several examples which show how some standard requirements such as freshness or agreement properties could be represented in this framework.

It is also possible to use strand spaces to provide a very convenient way of expressing a limited type of correspondence. Strands can be parameterized by the name of the principal executing the strand and the data it sends and receives. Thus, in the Station-to-Station protocol the initiator's strand would be parameterized by $\mathrm{Init}[A, B, X, Y, K]$, while the responder's would be parame-

terized by Resp$[B, A, X, Y, K]$, where X and Y are the initiator's and responder's Diffie-Hellman components, respectively, and K is the key derived from the Diffie-Hellman exchange. Earlier, we described how Lowe showed that after A completed the Station-to-Station protocol, A and B would agree on B's identity and on the Diffie-Hellman components and key, but not A's identity. We could express that fact as a requirement that, after an initiator A finishes executing the protocol, apparently with a responder B, if the initiator's strand is Init$[A, B, X, Y, K]$, then the responder's strand is Resp$[*, A, X, Y, K]$, where * denotes a wild card. Unlike the strand space pictures, this notation cannot express conditions on the relative times of occurrence of events from two different strands. However, since many requirements have to do not so much with agreement on placement of events as with agreement on data such as keys, this notation has been useful for in a number of different cases. It would be interesting to see how far it could be extended and still retain its compactness and readability.

A somewhat different approach has been taken by Cervesato and Meadows [7] in the development of a graphical representation of the NPATRL language. This representation was based on the fact that queries in the NRL Protocol Analyzer, for which NPATRL was designed, are couched in terms of events that should or should not precede some specified event. Such a way of formatting queries has an obvious connection to fault trees. A fault tree is a graphical means of representing failure modes in safety-critical systems. The root of the tree represents the failure with which the system designer is concerned, and the branches represent the conditions under which the fault can occur. The main difference between NPA queries and fault trees is that in NPA queries the relationship is one of precedence, while in fault trees it is one of causality. Otherwise the structure is very similar. Moreover, the graphical representation makes it easier to understand the relationships between the various events. For this reason, we found it very helpful, in particular, to represent the GDOI requirements, especially the more complex ones, in terms of fault trees. In [7] a fault tree semantics for the subset of NPATRL requirements accepted by the NPA is developed, and some sample requirements are shown.

5 Quantitative and Probabilistic Requirements

So far, with few exceptions, the requirements we have looked at have dealt with safety requirements for discrete systems. This fits well when we want to analyze authentication and key distribution protocols that follow the Dolev-Yao model, where the cryptosystem is a black box, and principals communicate via a medium controlled by a hostile intruder who can read, alter, and intercept all traffic. But, since the correctness of a protocol depends on the correctness of the cryptoalgorithm that uses as well as the way it uses those algorithms, it would be useful to have correctness criteria that took the properties of the cryptoalgorithms into account.

Prior to and concurrent with the explosion of formal methods approaches to cryptographic protocol analysis, there has been a parallel effort in developing correctness criteria for cryptoalgorithms and cryptographic protocols based on complexity-theoretic approaches. Indeed, the work of Bellare and Rogaway cited earlier was developed in such a context. What has been lacking however, has been a means of integrating such a complexity-theoretic approach with the logical systems that we have been considering in this paper. However, some work in this area is beginning to appear, such as the work of Abadi and Rogaway [2], which considers a complexity-theory based model as a semantics for a logical system, although it restricts itself to secrecy requirements, and the work of Mitchell et al [25], which develops a notion of bisimulation that takes into account complexity-theoretic and probabilistic considerations.

The use of cryptography is not the only place where quantitative requirements become relevant. For example, many anonymity protocols, are intended to provide a statistical notion of security. An intruder may have a nontrivial chance of guessing the identity of a sender or receiver of traffic, but we do not want that chance to exceed a certain threshold. Protocols intended to protect against denial of service attacks may need to limit the the amount of resources expended by a responder in the early steps of the protocol. Recently, researchers have begun to investigate ways of applying formal methods to the analysis of protocols that must satisfy quantitative requirements. Examples include the work of Meadows on a model for the analysis of protocols resistance to denial of service [21] where requirements are specified in terms of a comparison between resources expended by a responder versus resources expended by an initiator; the work of Buttyán and Hubaux [6] on rational exchange protocols, in which a protocol is modeled as a game in which all principals are assigned payoffs, and an exchange protocol is deemed rational if the strategies available to all participants form a Nash equilibrium; and the work of Shmatikov on anonymity protocols and contract signing, in which the protocols and their requirements are modeled in terms of Markov chains [29, 26], making them amenable to analysis by probabilistic model checkers.

6 Conclusion

We have given a brief survey of research in expressing cryptographic protocol requirements. We believe that it this point we have a good handle on the specification of the standard secrecy and correspondence requirements of security protocols. It appears possible to derive techniques that are compatible with just about any type of formal system, and we have a vast range of requirement specification styles, from one end of the extensional-intensional spectrum to the other.

There are of course a number of areas in which work on cryptographic protocol requirements needs to be extended. One is in making the requirements language user-friendly. Security protocols, and thus their requirements, can be complex; even more so when one must consider operation in partial failure modes

such as compromise of temporary session keys. Thus it makes sense to concentrate on ways of making requirements languages easier to use, even when the requirements are complex. In this paper we discussed some of the work on graphic requirements languages that attempts to address this problem.

Another area in which work is just starting is in extending cryptographic requirements specifications beyond secrecy and correspondence. These would apply to protocols whose goals go beyond those of key distribution and authentication that have traditionally been handled in this area. One area of particular interest here is quantitative requirements. We have pointed out some areas in which the ability to understand a protocol's behavior from a quantitative point of view appears to be crucial. In this case, not only requirements need to be developed, but formal models for specifying the protocols that must satisfy the requirements. We have described some of the work in this area as well.

There are some other areas which could also use more exploring. For example, many electronic commerce protocols must satisfy various types of non-safety requirements. Is it possible to develop ways of characterizing and specifying these requirements in ways that are particularly relevant to security protocols, as has been done for the safety properties of secrecy and correspondence? Another area of research has to do with interoperability. Increasingly, many protocols will rely upon other protocols to supply some of their security services. What is the best way to specify services needed by one protocol in terms of requirements upon another? We hope to see research in these and other emerging areas in the near future.

References

1. M. Abadi. Secrecy by typing in security protocols. *Journal of the ACM*, 46(5):749–786, September 1999.
2. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, to appear.
3. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO '93*. Springer-Verlag, 1993.
4. R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of two-party authentication protocols. In *Advances in Cryptology - Proceedings of CRYPTO 91*. Springer-Verlag, 1991.
5. Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, February 1990.
6. L. Buttyán and J.-P. Hubaux. Rational exchange – a formal model based on game theory. In *2nd International Workshop on Electronic Commerce (WELCOM'01)*, 16-17 November 2001.
7. I. Cervesato and C. Meadows. A fault-tree representation of NPATRL security requirements. submitted for publication, 2003.
8. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes, and Cryptography*, 2:107–125, 1992.
9. D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

10. F. J. Thayer Fábrega, J. Herzog, and J. Guttman. Strand space pictures. In *Proceedings of the Workshop on Formal Methods and Security Protocols*, 1998. available at `http://www.cs.bell-labs.com/who/nch/fmsp/program.html`.
11. F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171. IEEE Computer Society Press, May 1998.
12. R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In U. Montanari, editor, *27th International Colloquium on Automata, Languages and Programming (ICALP'00)*. Springer Verlag: LNCS 1583, July 2000.
13. Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In R. K. Iyer, M. Morganti, Fuchs W. K, and V. Gligor, editors, *Dependable Computing for Critical Applications 5*, pages 79–100. IEEE Computer Society, 1998.
14. J. Goquen and J. Meseguer. Security policy and security models. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.
15. A. Gordon and A. Jeffrey. Authenticity by typing in security protocols. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2001.
16. A. Gordon and A. Jeffrey. Typing one-to-one and one-to-many correspondences in security protocols. In *International Software Security Symposium (ISSS 2002)*. Springer LNCS, 2003.
17. Paul Hoffman. Features of proposed successors to IKE. Internet Draft draft-ietf-ipsec-soi-features-01.txt, May 31 2002. available at `http://ietf.org/internet-drafts/draft-ietf-ipsec-soi-features-01.txt`.
18. G. Lowe. Some new attacks on security protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society Press, 1996.
19. G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 31–43. IEEE Computer Society Press, 1997.
20. C. Meadows. Applying Formal Methods to the Analysis of a Key Management Protocol. *Journal of Computer Security*, 1:5–53, 1992.
21. C. Meadows. A cost-based framework for analysis of denial of service in networks. *Journal of Computer Security*, 2001.
22. C. Meadows and P. Syverson. A formal specification of requirements for payment in the SET protocol. In *Proceedings of Financial Cryptography '98*. Springer-Verlag LLNCS, 1998.
23. C. Meadows, P. Syverson, and I. Cervesato. Formalizing GDOI group key management requirements in NPATRL. In *Proceedings of the ACM Conference on Computer and Communications Security*. ACM, November 2001.
24. J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol Security Analysis. *IEEE Transactions on Software Engineering*, SE-13(2), 1987.
25. J.C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols (preliminary report). *Electronic Notes in Theoretical Computer Science*, 45, 2001.
26. G. Norman and V. Shmatikov. Analysis of probabilistic contract signing. In *BCS-FACS Formal Aspects of Security (FASec '02)*, 2002.

27. A. W. Roscoe. Intensional specification of security protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 28–38. IEEE Computer Society Press, June 10-12 1996.
28. S. Schneider. Security properties and CSP. In *IEEE Computer Society Symposium on Security and Privacy*. IEEE Computer Society Press, 1996.
29. V. Shmatikov. Probabilistic analysis of anonymity. In *Proceedings of the 15th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2002.
30. P. Syverson and C. Meadows. Formal requirements for key distribution protocols. In *Proceedings of Eurocrypt '94*. Springer-Verlag, 1994.
31. P. Syverson and C. Meadows. A formal language for cryptographic protocol requirements. *Designs, Codes, and Cryptography*, 7(1/2):27–59, 1996.
32. Paul Syverson and Catherine Meadows. A Logical Language for Specifying Cryptographic Protocol Requirements. In *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 165–177. IEEE Computer Society Press, Los Alamitos, California, 1993.