NRL Release Number 03-1221.1-0918

Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends

Catherine Meadows

***Abstract*—**
The history of the application of formal methods to cryptographic protocol analysis spans over twenty years, and recently has been showing signs of new maturity and consolidation. Not only have a number of specialized tools been developed, and general-purpose ones been adapted, but people have begun applying these tools to realistic protocols, in many cases supplying feedback to designers that can be used to improve the protocol's security. In this paper we will describe some of the ongoing work in this area, as well as describe some of the new challenges and the ways in which they are being met.

I. Introduction

The application of formal methods to cryptographic protocols begins with the analysis of key distribution protocols for communication between two principals. Alice and Bob want to talk to each other securely, and they need a session key in order to do that. They can either obtain it from a key server, or generate it themselves. Although this problem sounds simple, it is not always easy to solve, since any solution achieve its goals in the face of a hostile intruder who can intercept, alter, or delete messages, and may be in league with a certain number of dishonest principals. Thus, for a large part of its history, the field has concentrated on problems of this type.

However, as the world becomes more and more dependent upon computer networks to perform its transactions, and as cryptography becomes more widely deployed, the types of applications to which a cryptographic protocol can be put become more varied and complex, Examples include financial transactions, which may rely on properties such as liveness and fairness as well as the more traditional security properties guaranteed by cryptography; secure group communication, which requires a key to be kept secret within a group as members may join or leave; and negotiation of complex data structures such as security associations instead of keys. Moreover, the types of threats have become more varied. Now networks must defend not only against intruders who may attempt to learn secrets or impersonate honest principals, but they also must be robust against denial of service attacks, and, in some cases, against traffic analysis as well. Any attempt to develop a method to assure correctness of cryptographic protocols must take these new developments into account.

In this paper we will describe what we see as some as the emerging trends in cryptographic protocols, and describe some of the new problems they pose for their analysis. We will also describe some of the research that is going on in these areas. This paper is an expansion of and follow-on to an earlier paper [50] on the same topic. This gives us the opportunity to revisit many of the issues we have covered in that paper, and to reexamine many of the conjectures and predictions that we made.

The rest of this paper is organized as follows. In Section Two we give a brief history and survey of the state of the art in the field of the application of formal methods to cryptographic protocol analysis. In Section Three we describe what we see as some trends in protocol design that may affect the type of formal analysis that can be done. In Section Four we describe what we see as some of the emerging research areas arising out of these trends, motivated in many cases by our own experience in analyzing cryptographic protocols, and we describe some of the research that is being done in these areas. Section Five concludes the paper.

C. Meadows is Head of the Formal Methods Section at the Naval Research Laboratory, Washington, DC, USA (email: meadows@itd.nrl.navy.mil)

II. The History and Current State of Formal Cryptographic Protocol Analysis Tools

To begin with, it will help if we can give an idea of what we mean both by cryptographic protocols and formal methods.

Cryptographic protocols are protocols that use cryptography to distribute keys and authenticate principals and data over a network. The network is usually assumed to be hostile, in that it may contain intruders who can read, modify, and delete traffic, and who may have control of one or more network principals. A cryptographic protocol must be able to achieve its goals in face of these hostile intruders. Because of this, such protocols are often subject to nonintuitive attacks which are not easily apparent even to a careful inspector. Many of these attacks do not depend upon any flaws or weaknesses in the underlying cryptographic algorithm, and can be exploited by an attacker who is able to do no more than the basic operations listed above, but in any arbitrary order. Other attacks may depend upon subtle properties of the cryptographic algorithms, or on statistical analysis of message traffic, or on some combination of the above.

By "formal methods" we will mean, somewhat loosely, a combination of a mathematical or logical model of a system and its requirements, together with an effective procedure for determining whether a proof that a system satisfies its requirements is correct. Thus, we will not include analytical, reduction-based proofs, however rigorous, unless they can be shown to satisfy these criteria. On the other hand, we will include incomplete techniques such as finite-state model checking, since even if

though they do not give a proof of security for the entire possible state space, they do allow a precise statement of the conditions under which their conclusions hold, together with an effective procedure for checking them.

Until recently, the main body of work in formal methods has concentrated on analysis of discrete systems. Indeed, many security problems in protocol analysis can be formulated in terms of the properties of a discrete system. Several principals wish to exchange data securely, but are subject to attack by an intruder who can do any sequence of a finite set of operations such as intercepting data, concatenating and deconcatenating data, encrypting and decrypting data, and so forth. Since many protocol flaws can be exploited by such an attacker, and since these attacks are often nonintuitive, it appears that some short of discrete formal analysis would be helpful in helping us to avoid them. For this reason, it has long been realized that formal methods can be useful for the analysis of the security of cryptographic protocols. They allow one both to do a thorough analysis of the different paths which an intruder can take, and to specify precisely the environmental assumptions that have been made.

Probably the first mention of formal methods as a possible tool for cryptographic protocol analysis came in Needham and Schroeder [60]. However, the first work that was actually done in this area was done by Dolev and Yao [23], and slightly later by Dolev, Even, and Karp [22], who in the late seventies and early eighties developed a set of polynomial-time algorithms for deciding the security of a restricted class of protocols. Unfortunately, it was soon found that relaxing the restrictions on the protocols even slightly made the security problem undecidable [26], and so the work did not go much further than that. Dolev and Yao's work was significant, however, in that it was the first to develop a formal model of an environment in which multiple executions of the protocol can be running concurrently, in which cryptographic algorithms behave like black boxes which obey a limited set of algebraic properties (e.g. the encryption and decryption operations cancel each other out), and which includes an intruder who can read, alter, and destroy traffic, and may also control some legitimate members of the system. Most later work on the formal analysis of cryptographic protocols is based on this model or some variant of it.

Shortly later, work began on developing tools for the analysis of security protocols in general, all of which were based on the Dolev-Yao model or some variant, including the Interrogator [56], the NRL Protocol Analyzer [46], and the the Longley-Rigby tool [42]. Others applied general-purpose formal methods to the problem [39]. Most of this work used some type of state exploration technique, in which a state space is defined and then explored by the tool to determine if there are any paths through the space corresponding to a successful attack by the intruder. Inductive theorem proving techniques were also included in the tool in some cases, as in the NRL Protocol Analyzer, to show that the size of the search space was sufficient to guarantee security. Even during these early stages, much of this work was successful in finding flaws in protocols that had been previously undetected by human analysts, including the use of the NRL Protocol Analyzer to find a flaw in the Simmons Selective Broadcast Protocol [46], and the use of the Longley-Rigby tool to find a flaw in a banking protocol [42].

However, this still remained a fairly esoteric area until the publication of the Burrows, Abadi, and Needham logic [12] brought the problem to the attention of a larger research community. BAN logic uses an approach very different from that of the state exploration tools. It is an example of a logic of belief, which consists of a set of modal operators describing the relationship of principals to data, a set of possible beliefs that can be held by principals (such as a belief that a message was sent by a certain other principal), and a set of inference rules for deriving new beliefs from old ones. An example would be a rule saying that if A believes that a key K is known only by him and B, and A sees a message encrypted with K, than A believes that that message was sent by B to A, or by A to B. The BAN logic consisted of a very simple, intuitive set of rules, which made it easy to use. Even so, as the BAN paper demonstrated, it was possible to use the logic to pinpoint serious flaws in protocols. As a result, the logic gained wide attention and let to a host of other logics, either extending BAN logic or applying the same concept to different types of problems in cryptographic protocols.

Note that belief logics such as BAN are generally weaker than state exploration tools since they operate at a much higher level of abstraction. Thus interest in them has waned somewhat as state exploration systems have improved. However, they have an advantage in that they are usually decidable and often even efficiently computable, and thus can be completely automated, as has been shown by Brackin's Automated Authentication Protocol Analyzer [10].

More recently, research has focused on state exploration tools and theorem proving techniques based on the Dolev-Yao model, much of it sparked by Lowe's demonstration that it was possible to use a general-purpose model checker, FDR, to find a man-in-the-middle attack on the Needham-Schroeder public key protocol [43]. Lowe was not the first to suggest the application of a general purpose model-checker to this problem (the idea of using FDR and the CSP language upon which it is based was first suggested by Peter Ryan in 1994, and the first published work on this approach is Roscoe's in [68]), but Lowe's demonstration of the attack inspired many both to prove that they could reproduce his results and to apply their own techniques to other protocols. Work since then has progressed in applying both model checkers [58], [20] and theorem provers [63], [25] to the problem, as well as in the design of special-purpose model checkers [36], [74], [45] and the use of specialized tools originally intended for somewhat different applications [24].

Although model checkers can only search a finite number of states, there has also been active research in showing under what circumstances checking a finite number of states might be sufficient. Probably the first result in this area was a result of Lowe's giving a set of conditions under which checking a small number of sessions would be sufficient to prove secrecy of a key [44]. More recently, a sizable body of research has concentrated on developing model checkers which assume a bounded number of sessions, but come with a proof of completeness otherwise, without making any restrictions on message complexity such as depth of encryption. Huima's model-checker [36] is probably the first of this type; others to apply or extend this

approach include [74], [45], [5], [28], [55]. Recently Rusinowitch and Turanic [69] showed the secrecy problem to be NP-complete under these assumptions.

There has also been a sign of consolidation in the area, an indication that it has been maturing. For example, Millen has been developing CAPSL [21], the Common Authentication Protocol Language, which is intended to provide a common specification language for cryptographic protocol analysis tools. More recently, Thayer, Herzog, and Guttman [80] developed a graph-theoretic interpretation of the Dolev-Yao model, called the strand space model, that brings together many ideas and techniques that have been used in the formal analysis of cryptographic protocols. Because of this, and because of its simplicity and elegance, it has begun to be used both as a basis for new special-purpose tools [74] and as a framework in which to express theoretical results [75]. This trend has promising implications for the integration of future tools and the incorporation of new theoretical results into these tools.

Recently, we have also seen advances in the use of other types of tools besides model checkers in the analysis of cryptographic protocols. For example, Paulson has been able to use the theorem prover Isabelle to analyze protocols of significant complexity. Although use of a theorem prover is by nature more interactive than use of a model checker, Paulson has been able to build up a library of theorems and techniques that not only can be reused in other Isabelle analyses, but have been picked up and used by other researchers as well [54]. Others [19], [34] have developed special-purpose theorem provers and algorithms that are fine-tuned for cryptographic protocol analysis. These typically require much less user interaction than a general-purpose theorem prover, but provide greater coverage than model checkers, without the necessity of developing separate abstractions. Unlike model checkers, however, they do not provide counterexamples when they fail to prove security.

Probably the newest approach to formal cryptographic protocol analysis is the use of type checking [1], [31]. In type checking, messages and channels are assigned different types, and security flaws are identified as type violations (e.g., a data item of type private appearing on a channel of type public). Type checking has the advantage that, like model checking, it is completely automatic, but that, unlike model checking, it can handle certain classes of infinite systems. It has the potential disadvantage, though, that since security violations are defined in terms of type inconsistencies, the security requirements to be proved must be considered when the specification is being written. This is in contrast to the use of model checkers, for which any security property that can be expressed in terms of temporal logic can be specified independently after the protocol itself is specified. Since this approach is so new, it is unclear how it will ultimately work out, but it is one that bears watching.

To sum up, at present, as we predicted a few years ago, the field has now reached a state in which there are a number of different tools available that can be used to verify safety properties such as authentication and secrecy by performing an analysis of a protocol specified at the same level of detail that is normally provided in a journal paper using the Dolev-Yao model of a protocol attacker, using techniques ranging from state space exploration to theorem proving to type checking, and possibly beyond. This is not everything. First of all, such tools will not catch errors that arise from implementation details that go beyond the type of specification one might see in a high-level description such as one might find in a journal paper. Secondly, the Dolev-Yao model leaves out some important attacker capabilities such as cryptanalysis. Finally, since the Dolev-Yao model assumes an intruder that is capable of blocking any message, it is impossible to prove any kind of liveness property. Nevertheless this is still a great deal; the wide range of attacks found by such tools demonstrate that a number of nontrivial problems occur at this level of specification.

Since the protocol security problem is undecidable, [26], [35], [15], the analysis tools will not be successful all the time, and they may require human intervention at times. But even so, the problem of tool design for this area seems well enough understood by now so that these limitations should not interfere too much with their effective use.

Given that we have reached this plateau, it seems reasonable at this point to ask, what comes next? And indeed, there are a number of related problem areas still to be explored. Some of them have only surfaced in the last few years. Others have been known about for some time, but it was thought more important to concentrate on the basics first. However, now that the basics are well understood, it is time to look more closely at some of these areas. In the remainder of this paper we do this, in each area pointing out what work has already done, and what we believe still remains to be done.

III. Trends in Cryptographic Protocols

In this section we describe what we see as some of the emerging trends in cryptographic protocols. Since these trends present new challenges to protocol analysis, we will use this section to motivate our discussion of what we believe to be some of the more pressing open problems.

a) Greater Adaptability and Complexity: Probably one of the most obvious trends is the increasing different kinds of environments that protocols must interoperate with. As networks handle more and more tasks in a potentially hostile environment, cryptographic protocols take on more and more responsibilities. As networking becomes more widespread, and different platforms must interoperate, we see protocols such as the Internet Key Exchange (IKE) protocol [33], that not only must agree upon encryption keys, but on the algorithms that are to use the keys. Or, we may see protocols such as SET [71] that must be able to process different types of credit card transactions.

One way of attempting to meet this challenge is to increase the complexity of the protocol. This of course, not only makes verification but implementation more difficult as well, and as a result there is always resistance to this approach. For example, the complexity of SET has had much to do with the general reluctance to adopt it, and dissatisfaction with the complexity of IKE resulted in the decision by the Internet Engineering Task Force to scrap it and start again from scratch. However, the tendency to greater complexity will always be there, and it will ultimately have to be met at least part of the way by anyone who is attempting to perform any type of security analysis.

b) Adoption of New Types of Cryptographic Primitives: In general, it is accepted that a conservative approach to algorithms is best when designing cryptographic protocols; only tried and true algorithms should be used. But, as the field matures, the number of algorithms that are considered to have received enough scrutiny has increased. Moreover, as computing power increases, algorithms that were once considered prohibitively expensive have become easier to implement, while others, such as DES, are widely regarded as no longer providing adequate security.

However, many of these algorithms possess properties that are not modeled by existing protocol analysis systems. Most of these systems use a very simple model of encryption; a principal who knows a key and a message can produce an encrypted message, and a principal who knows an encrypted message and the corresponding decryption key can produce the original message. This is not adequate to model something like the Diffie-Hellman algorithm, which depends, as a minimum, on the commutative properties of exponentiation, or something like Chaum's blinded signatures, which depend upon the homomorphic properties of RSA. Other algorithms and data structures, such as Chaum mixes [17], designed for preserving anonymity in networks, or hash trees, which have found new application in group key management protocols, do not introduce new algebraic properties, but the fact that they are of unbounded size means that it is not obvious how to reason about them with existing protocol analysis systems. Finally, other primitives make use of time in non-trivial ways, for example the TESLA protocol's use of timestamps to implement multicast packet authentication [66].

Finally, as theoretical cryptography matures, we are beginning to see the introduction of practical cryptographic algorithms that have their own proofs of security, e.g. by reduction to hard problems such as factoring. It would be helpful if any analysis of protocols that use these cryptographic algorithms could also make use of their security proofs, so that any proof of the protocol's security would be valid right down to the algorithm level.

c) New Types of Threats: In the early years of computer security, much of the threat analysis was hypothetical, and focused on attacks in which there would be a clear (usually monetary) gain for the attacker, such as fraud or compromise of secrets. Now, with more experience, we see that there are other types of attacks, most of them related to denial of service, that can prevent a network from fulfilling its functions. Many denial of service attacks can be countered by good resource management (e.g. shutting off parts of a system that are under attack, reallocating resources to parts that are not under attack). But sound protocol design can do much to help, for example by keeping a responder from committing its resources to communicating with an initiator until it has adequate assurance that it knows who it's talking to. This can be a delicate problem, however, since many of the techniques used for authentication themselves require commitment of resources, and since the decision of how much resources to commit, and when, can be very implementation-dependent. Successful analysis will depend to some extent on the ability to compare the resources expended by an attacker to the resources expended by a defender.

Other threats, such as traffic analysis, focus on problems that are not really an issue until adequate cryptographic protection for communication secrecy has already been attained. Protection against traffic analysis is one of these. Even when encryption is used, source and destination of message traffic is not hidden, and it can be possible for an observer to learn much from this alone. Recently, a number of different systems have been developed that attempt to solve this problem with varying degrees of completeness. However, without some ability to evaluate and compare the degree of protection offered by these systems, it is difficult to assess what amount and kind of security they offer. Such analysis methods should take statistical techniques into account, since much traffic analysis depends on statistical analysis. Ideally, they should also take into account the resources and communication capabilities of intruders, since successful traffic analysis depends on the correlation of data taken from several sources.

A somewhat different type of threat emerges when we look at electronic commerce protocols. In this type of protocol, the parties involved participate in a transaction that results in certain levels of payoff to each principal involved. Moreover, the protocol may either depend upon or try to guarantee liveness or fairness properties as well as safety properties. A principal may try to cheat by trying to increase its payoff at the expense of those of other parties, but will not engage in behavior that will result in a lowering of its payoff, or put it at a disadvantage with respect to the others. This is in contrast with the Dolev-Yao style of protocol analysis, where the intruder is assumed to be willing to engage in any behavior that will disrupt the correct execution of the protocol, where the properties proved are safety properties such as authentication, and where liveness properties cannot be depended upon or guaranteed, since the intruder is assumed to have complete control of the network. Thus, in order to assess the security of a protocol of this type, both the protocol requirements and the intruder will have to be modeled differently.

IV. Emerging Areas of Research

A. Open-Ended Protocols

Most of the work on the formal analysis of cryptographic protocols has concentrated on protocols that involve the communication of a fixed number of principals: for example, an initiator and a responder in a key agreement protocol, or a customer, merchant, and bank in an electronic commerce protocol. Most data structures that are used are also closed-ended. That is, each message is a fixed structure that is composed of a fixed number of fields containing data such as nonces, names, keys, etc. Open-endedness is included in the protocol model, but only with respect to the number of protocol executions that may be going on at the same time, or the number of operations that the intruder may perform to create a message. This means that the models do not need to include such constructs as loops, thus simplifying the model and, one hopes, the analysis.

However, open-ended structures are beginning to show up in a number of different applications. By open-ended, we simply mean the the structure may include an arbitrarily large number of data fields; either no precise limit is put on them by the protocol specification, or the bound is so large that for the purposes

of analysis we may as well assume that it does not exist. One example of an open-ended structure is in group communication protocols, in which keys must be shared among members of a group of arbitrary size. Here, it is the group of principals that may be participating in a particular instance of the protocol that is open-ended. However, open-ended structures show up in other types of protocols, as well. For example, anonymous routing protocols make use of an arbitrary number of routers to achieve their goals.

Open-ended structures are also used even in protocols in which the number of principals is bounded. For example, the SET protocol allows a merchant to batch transactions for approval by a security gateway. The IKE Protocol offers an even more complex example. One of the purposes of IKE is to agree on a security association (SA). A security association is the collection of algorithms and other information used to encrypt and authenticate data. Although there is some information that an SA must include, there is no defined limit on what it can include, so its definition is left open-ended. In addition, an SA is negotiated by having the initiator present a list of SAs to a responder, who then picks one. Thus there are two sources of open-endedness in the use of SAs. Moreover, this open-endedness is security-relevant. For example, Zhou [86] and independently Ferguson and Schneier [27] found an attack in which an intruder could trick an initiator into agreeing on the wrong SA by making use of the fact that only part of the SA is actually used in IKE itself. More recently, candidate successors to IKE have offered SAs in which a given suite may, for example, offer the option of a number of different types of a cryptographic algorithm, resulting in potentially more complex, but also more compact, SA specifications.

Work on the analysis of these types of open-ended structures is slowly beginning to appear, mostly in the area of group protocol analysis. Probably the earliest work on group protocol analysis was Paulson's [61] application of the Isabelle theorem prover to the analysis of a protocol that involves an arbitrary number of principals, and the work of Bryans and Schneider [11] applying the PVS theorem prover to the same protocol. Since all of this work involves general-purpose theorem provers instead of special-purpose tools, we would not be surprised to find that the authors were able to make use of techniques that were not available in tools that were specifically designed for cryptographic protocol analysis. However, it is heartening to note that Bryan and Schneider's work makes use of a construct, the rank function, that Schneider had previously developed for the analysis of cryptographic protocols that involved only a bounded number of participants in a single protocol execution.

Other work concentrates on the analysis of a version of Tsudik's Cliques protocol, which is based on group Diffie-Hellman, in which a principals pass a modulus among themselves, each raising it to a secret exponent. This, not only is the protocol open-ended, but so is the cryptographic primitive that it relies upon. This and the protocol's simplicity makes it a good case study. A formal analysis with the unification-based NRL Protocol Analyzer was done in [49]; this used an ad-hoc unification algorithm that was not guaranteed to terminate, and was able to prove that if the group never contains any dishonest members, then secrecy of the key is preserved. Later, Meadows and Narendran developed a unification algorithm for group Diffie-Hellman that could be used for such an analysis [51]. Pereira and Quisquater have also performed an analysis of the same protocol be reducing the secrecy problem to a problem in linear algebra; they were able to demonstrate that a secret key can be compromised by a former member, even if it no longer belongs to the group [65].

More recently, Meadows et al. have been concentrating on an analysis of a group protocol being developed as a standard by the IETF, the Group Domain of Interpretation (GDOI) protocol (see [52] for a preliminary report). This is structured more like a conventional protocol than the others that were analyzed above, since in GDOI the key is generated and distributed by a group controller instead of the entire group. The protocol consists of two parts. In the first part, a group member requests to join the group and obtains a key from the group controller. In the second part, the group controller sends a key update message whenever it wants to send a new key to the group. The main source of unboundedness here (besides the hash trees that are used to update keys, which were not modeled in this analysis), is the fact that a single session can be arbitrarily large, with an arbitrary number of group members joining and leaving. This is handled by using the NRL Protocol Analyzer's ability to recognize when a state is essentially "identical" to a state that it has already encountered and thus not necessary to explore further, as well as the ability on the part of the user to control the portions of the state space searched by the NPA so that the validity of a proof of security is not affected. The techniques used are basically the same as those that the NPA already used, namely a combination of the use of variables to represent message terms and principal names, and partial order reductions similar to those used by Clarke et al. [18].

To sum up, at this point we appear to have a combination of general techniques that can be scaled up to deal with these unbounded systems, as well a scattering of more specialized techniques that have been developed for dealing with particular problems that could not be handled by the more general techniques. However, as people continue to explore these types of problems, we expect that a more systematic approach may emerge.

B. New Applications and Threats

Earlier in this paper, we talked about emerging applications of cryptographic protocols: electronic commerce, protection against denial of service, and anonymity and resistance to traffic analysis. All of these have the property that the threat model and goals of the intruder are somewhat different from that of the traditional Dolev-Yao model. They also have the property that reasoning about them could in many cases by helped by an ability to reason about quantified as well as symbolic information. In the case of denial of service, there is the necessity of estimating the amounts of resources expended by a principal when participating in a protocol. In the case of anonymity, there is the necessity of estimating the statistical information about source and destination that an opponent could learn by observing or interfering with message traffic. Finally, for many of the electronic commerce and contract signing protocols, it is

possible to put a quantitative value on the payoff to the various parties. We will consider each of these problems in more detail.

1) Denial of Service: Denial of service was not a threat that was a cause of much concern to the first designers of cryptographic protocols. However, as we have seen from the SYN attacks on TCP/IP, many communication protocols are subject to a particular type of denial of service attack in which the attacker initiates an instance of a protocol and then drops out, leaving the victim hanging. Since the victim must use resources to keep the connection open until the protocol times out, the attacker, by initiating and then dropping enough instances in the protocol quickly enough, can cause the victim to waste enough resources keeping connections open so that it is unable to participate in any more instances of the protocol and is thus effectively cut off from the network.

Strong authentication can both ameliorate and exacerbate this problem. Authentication can be used to identify the source of the attack, allowing the victim to cut off communication with the attacker. But authentication can also be used as a means of launching denial of service attacks, since it is both computation and storage-intensive, and the attacker could launch a denial of service attack on a victim by sending it a series of incorrectly authenticated messages that it would waste its resources verifying.

The approach that has been taken to resolving this problem is to use a tradeoff between resources required of the victim (referred to from now on as the "defender") with resources required of the intruder. Early parts of the protocol require weak authentication that do not require great resources on the part of the intruder to break, but require fewer resources on the part of the defender to verify. More expensive forms of authentication are reserved for later in the protocol when a degree of assurance that the participating parties are legitimate are obtained. This, for example, is the strategy behind the use of cookies, as originally proposed in the Photuris protocol [37]. A cookie exchange provides a weak form of authentication that does not require much commitment of resources. Once this is performed, the principals can proceed to use stronger, more expensive, authentication.

Note that the attacker model used in this strategy is generally weaker than the model used in the verification of traditional authentication goals. Thus the sort of nonintuitive attacks that have been found on these types of goals will not necessarily arise in the case of denial of service, although they are not ruled out entirely. However, the analysis becomes more complicated in a number of other ways. First, the protocol must be analyzed, not only in terms of its final goals, but along each step of the way. Every time a a principal takes part in some action that requires the use of a significant amount of resources, one must check that that an attacker could not fraudulently cause that principal to reach that step without spending a significant amount of its own resources. Secondly, in order to make that verification possible, it is necessary to have a model, not only of principal and intruder actions, but of the cost of those actions. Thus some sort of formal analysis technique would be beneficial, simply in order to to keep track of this complex multi-stage analysis.

Existing protocol analysis tools, although they cannot be applied to the problem directly in their present form, have many features that could be useful if adapted properly. For example, for most it is possible to specify intermediate as well as ultimate goals. Also, although most use a single model of the intruder, most of the weaker intruder models that would be used would be restrictions of this more general intruder model.

Our own work has concentrated on a framework [48] that could be used to apply existing tools, appropriately modified, to the denial of service problem. We make use of the concept developed by Gong and Syverson [30] of a *fail-stop* cryptographic protocol. Briefly, a protocol is fail-stop if, whenever an attacker interferes with a message, this is detected by the receiving principal and the protocol is halted. We have modified the fail-stop model to include an attacker whose capabilities change as the protocol progresses, and have developed a framework for trading off intruder capabilities against effort expended by the defenders. In this framework the protocol designer specifies a *protocol tolerance relation*, which describes how much effort he believes it should be necessary to expend against an attacker of a given strength. Since we are developing a framework for models instead of a specific model, we do not specify exactly how this effort should be quantified, but examples would include amount of resources expended, amount of time expended, or amount of computational power required. A protocol is then designed so that the effort expended by the defender increases as the protocol executes, and also as each message is verified. The protocol is then analyzed to show that it is fail-stop against an attacker whose capabilities are within the constraints of the desired tolerance relation. That is, at each verification point, the amount of effort required by the attacker to spoof the verification, versus the amount of effort wasted by the defender if the verification is successfully spoofed, should fall within the tolerance relation.

Since then, we have seen little work on applying formal methods to these types of problems. This may be partly because much of the analysis requires a level of implementation detail that goes much beyond what is usually offered by Doley-Yao style specification, partly because of the challenge of integrating discrete and quantitative analysis, and partly because most research recently has concentrated on areas such as traceback and resource management. However, denial of service is definitely something that is being taken into account by today's protocol designers, and anything that provides assistance in doing this in a systematic way should be helpful. Thus we believe that we will be seeing more work on this in the future.

2) Anonymous Communication: Anonymous communication is an application that has recently begun to move from the laboratory to the real world, as the ubiquity of the World Wide Web makes even ordinary users more sensitive to the dangers of traffic analysis and of indiscriminately revealing personal information over the Web. Thus systems such as the Onion Router [29], the Anonymizer and Crowds [67], are designed to prevent an onlooker from determining the origination or destination of requests to servers. Basically the way all these systems work is by having requests from users routed through one or more nodes. In the simplest versions, a user proxies a request through a single site, which strips the request of identifying data or otherwise disguises its source, and forwards it to the server.

More sophisticated systems, such as Crowds and Onion Routing, have the request routed through a number of nodes. Onion Routing uses cryptographic means to keep each node ignorant of all other nodes in the path except the ones with which it communicates directly.

We note that anonymity has properties that make it challenging to analyze. First of all, it is an emergent property, that is, one that is not apparent in isolation. It would be hard to disguise the source of a request if it is the only request in the network, no matter how many nodes it was routed through. An anonymizing protocol depends upon a mix of traffic to disguise the source and destination of any particular item. Statistical studies, such as the work of Timmerman [82] or Wright et al. [85], will be of use here to determine how well this strategy works in different situations. Previous work done on the analysis of covert channels in networks [84] might also be of use here, if applied properly. Next, the more powerful anonymizing protocols require communication between an arbitrary number of nodes, instead of just two or three, which is the number of principals that are usually assumed to be communicating in most of the protocols that have been analyzed using the existing cryptographic protocol analysis tools. Thus any techniques that are developed for analyzing group communication protocols will probably also be useful in this area.

We note that also that anonymity can in many ways be thought of as a property similar to information flow in multilevel systems in that, in information flow the system should look the same to Low no matter what actions are being performed by High, while in anonymity the system should look the same to the intruder no matter who is carrying out the communication. This insight was originally used by Merritt in his use of invisible homomorphism [53], and a similar insight was used later by Schneider and Sdirodoulos [70] in a CSP analysis of Chaum's Dining Cryptographers protocol.

Finally, since an anonymity protocol attempts to preserve secrecy by distributing the data over a wide area, the assumptions made about the capacities of an attacker are very relevant. A ubiquitous attacker will be able to break most anonymity protocols, but ubiquity is not always a very realistic assumption. However, attackers who only reside at single nodes and do not communicate are not very realistic either. Work by Syverson and Stubblebine [78] deals with some aspects of this problem, introducing the notion of a group principal that consists of a certain number of members who are assumed to have certain specified capacities for sharing knowledge.

The above types of analyses can be carried out with standard tools. But we note that there are also tools beginning to be available that allow one to combine formal methods with statistical analysis. In particular, Shmatikov has used a probabilistic model checker to construct several attacks on the Crowds system involving the reconstitution of paths [72], one of these similar to one of the attacks discussed in [85]. It would be interesting to see if an approach like this could be combined with some of the more traditional approaches described above, for example to suggest theorems that could be proved about an anonymizing system.

In summary, we believe that the analysis of anonymity protocols pose new research challenges which are beginning to be met, at least partially and in different aspects. The main challenge may be tying these different threads together.

3) Electronic Commerce: Most work on model-checking cryptographic protocols has been done to verify safety properties that can be expressed in terms of conditions on system traces. However, many desirable properties of electronic commerce protocols, such as fairness, cannot be expressed in this way. However, it is still possible to use model checkers designed for checking safety properties to analyze at least an approximation of the safety and liveness properties. Indeed, this was done by Shmatikov and Mitchell in [73], in which several attacks are found on published contract-signing protocols, including ways in which, for one protocol, a malicious principal can produce inconsistent versions of the contract and mount a replay attack, and for another protocol, the trusted third party is able to allow abuse or unfairness without being held accountable by the cheated party. However, as Shmatikov and Mitchell themselves point out, not all the desirable properties of such protocols are always amenable to analysis by their model-checker, Murϕ, which checks safety properties. Since properties such as fairness and abuse-freeness that are intended to be achieved by these protocols are not safety properies, it was necessary to approximate them by properties that Murϕ could check.

Since then, others have approached the problem in a more direct way. In [16] Chadha et al. express fairness properties of contract signing protocols in terms of balance properties of the entire execution tree, instead of properties of individual traces. Although this kind of property cannot be model-checked easily, they show how it can be proved inductively, giving an example for the Garay-Jakobsson-Mackenzie two-party contract-signing protocol. In [40] Kremer and Raskin deal with the problem by using a model-checker, MOCHA, that is based on a game semantics to check for fairness and timeliness in non-repudiation protocols. This semantics gives a more natural way of expressing these properties, which depend upon liveness as well as safety, than a trace-based semantics. More recently, Buttyán and Hubaux [14] have developed a game-theoretic model for exchange protocols, in which a protocol is modeled as a game in which strategies for principals are assigned payoffs. They then formalize the notion of rational exchange: an exchange protocol is deemed rational if the strategies available to the parties form a Nash equilibrium, so that no principal can increase its payoff by deviating from the rules of the game. They then show how their notion relates to the more traditional notion of fair exchange. Thus, we can already see some encouraging results in this area with tool support or the promise of it.

C. High Fidelity

Most work on the application of formal methods to cryptographic protocol analysis have modeled protocols at a very high level of abstraction. Techniques based on state reachability analysis usually assume that the algorithms used behave like black boxes, with only enough algebraic properties included (e.g. that encryption and decryption cancel each other out) to allow the protocol to function correctly. Techniques based on belief logics are usually even more abstract, forgoing in most cases even a general explicit model of the intruder or of the

cryptographic operations; instead goals achieved by the protocol are derived directly from the messages sent.

However, it is well known that many security problems in protocols arise at a much lower level of abstraction. Some come from interactions of the cryptographic algorithm with the protocol, such as a protocol that includes known or chosen plaintext while using a cryptographic algorithm that may be vulnerable to attacks based on the inclusion of this type of plaintext. Others come from problems with other supporting algorithms, such as hash functions or modes of encryption. Some come from other types of low-level implementation details. For example, in our analysis of the Internet Key Exchange protocol, we found an attack that would work if a recipient's decision as to the possible source of a message was implemented in one way, but would fail if it was implemented in another way (see [47] for a discussion of this). Thus, it appears to be well worth our while to take our analysis to a lower level of abstraction.

Some work in this direction already exists. For example, work on the analysis of modes of encryption and chosen and known plaintext has been successful both in finding new problems [76] and reproducing known attacks [77]. Work also exists on extending standard protocol analysis techniques to include Diffie-Hellman, including belief logics [83], [79] and model-checking techniques [47], [64].

Another approach is to attempt to wed formal methods with the proofs of security provided by theoretical cryptography, thus obtaining, not only an automated state analysis, but a rigorous proof of security based on well-defined cryptographic assumptions. These cryptographic theories generally are based on ideas from complexity theory and probability; that is, an opponent whose computational powers are limited in some way (e.g. to the ability to perform probabilistic polynomial-time computations), can break the algorithm with only negligible probability, where "break" could be defined in a number of ways, depending upon the algorithm's goals: for example, the ability to tell, given two ciphertexts, whether they have the same plaintext, the ability to produce two messages that hash to the same value, and so forth. The approach that has been taken is to develop both a computational model and a formal system so that the formal system can be proved sound with respect to the model. Examples include that of Lincoln et al. [41] and Abadi and Rogaway [2]. An advantage of this approach is that one can design the formal system so that it is amenable to checking by currently available verification techniques, while the underlying model can express the more detailed computational approach.

Thus we see new interest in the problem of high fidelity from a number of different sides, both from the point of view of modeling particular low-level properties of cryptographic algorithm, and the point of view of developing security proofs based on computational theories of cryptographic security.

D. Composability

Most work on the analysis of cryptographic protocols has concentrated on the analysis of protocols that can be described in terms of a single sequence of messages without any choice points or loops. However, most cryptographic protocols are not actually deployed in this fashion. Indeed, many cryptographic protocols as they are actually implemented can be thought of as a suite of "straight-line" sub-protocols (that is protocols that involve no if-then-elses and no loops) along with a number of choice points in which the user may choose which sub-protocol to execute. In this kind of environment, it is necessary, not only that each subprotocol be shown to execute correctly in isolation, but that the subprotocols do not interact with each other in harmful ways. This problem in its general form is known as the composition problem for cryptographic protocols: given that two or more different protocols are executing in the same environment, is it possible that a message or messages from one protocol could be used to subvert the goals of the other?

The composability problem is not only a theoretical concern. Consider, for example, the following attack, described in [8] on a very early version of SSL. The early version included an optional client authentication phase in which the client's challenge response was independent of the type of cipher negotiated for the session, and also of whether or not the authentication was being performed for a reconnection of an old session or for a new one. Moreover, this version of SSL allowed the use of cryptographic algorithms of various strength (weak algorithms for export and stronger ones for home use), and since weakness could be guaranteed by revealing part of the key, it was not always clear by inspection of the key whether weak or strong cryptography was being used. This allowed the following attack (note that in this version of SSL, session keys were supplied by the client):

1) A key K is agreed upon for session A using weak cryptography.
2) Key K is broken by the intruder in real time.
3) The client initiates a reconnection of session A.
4) The intruder initiates a new session B, pretending to be the client, using strong cryptography together with the compromised key K.
5) As part of the connection negotiations for session B, the server presents a challenge to the client. The client should return a digital signature of both K and the challenge. The intruder can't do this itself, but it can pass the server's request on to the client, who will take it to be part of the reconnection negotiations for session A, and produce the appropriate response. The intruder passes the response on to the server as part of the session B negotiations, and the protocol completes.
6) If the client would have been given access to special privileges as a result of using strong cryptography, this could lead to the intruder gaining privileges that it should not be able to have by breaking the key K.

Since this attack involves a confusion of the reconnection protocol with the connection protocol, it is an example of a failure of composition which would not have been found if the two protocols had been analyzed separately.

Other attacks involving interactions between protocols can be found in [3], [4].

Although these problems do not necessarily occur as often as other types of flaws, they are prevalent enough so that, in order to show that a collection of protocols is insecure with any degree of confidence, it is necessary to show that they are free from insecure interactions. Even worse, it has been shown that it is impossible to design a protocol that is secure against all

interactions with other protocols; that is, given a protocol, it is always possible to construct a protocol that can be used to "attack" it [38], [4]. This has a serious practical impact on the verification of cryptographic protocols; in order to show that a collection of protocols is secure, it is necessary to demonstrate that no protocol in the collection will accept a message sent by another protocol in the collection.

Early work on composability such as that of Heintze [35] and Gong and Syverson [30] concentrated on determining under what conditions protocols could be guaranteed to be composable with each other. The early results led to rather stringent requirements: in essence, they required the fail-stop property [30] or something very similar to it [35]. Thus they did not have much practical application.

More recent work has concentrated, not on designing protocols that are guaranteed to be composable with each other, but on reducing the amount of work that is required to show that protocols are composable. In our work in using the NRL Protocol Analyzer to analyze the Internet Key Exchange protocol, we found it useful to take each state transition that required an input message and determine which transitions could produce that output. This information was stored in a database, and only those rules that have a chance of producing that output are consulted when the reachability of an output is being verified. This allowed us to reduce the number of state transitions that had to be tested whenever we had to determine how a message could be produced. We do not make any claims for the originality of this idea (indeed some sort of rule pre-verification and storage is normally done by rule-based systems), but we were surprised at the dramatic speedup it caused, (at least threefold for the IKE analysis [47]). This is a technique that would be useful for the analysis of any complex protocol, but which was particularly helpful for the analysis of suites of protocols, since only a few state transitions from different protocols had the potential of interacting with each other.

Independently, Thayer, Herzog, and Guttman used a similar insight to develop a technique for analyzing composition properties using their strand space model. Their technique consists of showing that a certain set of terms generated by the first protocol can never be accepted by principals executing the second protocol. This information is then used to prove the full result that the first protocol does not interfere with the second. In [81] the authors show how this general strategy can be applied to specific protocols. In [32] Guttman and Thayer prove more general results that will guarantee composability: namely that, with certain restricted exceptions, the protocols do not use the same ciphertext as part of any message, and that the protocols satisfy certain conditions on the revealing of encrypted data.

The work of Thayer, Herzog, and Guttman provides a useful framework to approach the problem of composability, but there are still a number of open questions left. In particular, it is useful to ask how far we can relax their assumptions (for example, by allowing the compromise of old session keys), and still be able to prove useful results. It may also be necessary to look very closely at the question of whether or not two messages can be confused with each other. For example, in our own analysis of the Group Domain of Interpretation Protocol, we found a case in ambiguous formatting which a signed message intended for one subprotocol could be confused with a signed message intended for another. The confusion occurred at a level of detail not usually modeled in formal analysis of cryptographic protocols, but even so, a framework such as that provided by the strand space model might be helpful in identifying where such confusions are likely to occur.

E. Getting it into the Real World

Throughout most of this paper, we have concentrated on extending the limits of research. But we also need to concentrate on getting the results of our research out to the people who can make best use of it: the designers and evaluators of the cryptographic systems that are being deployed in our networks. For this we want to concentrate not on cutting-edge research problems, but on what we do best now, and what is the best use we can make use of these capabilities.

We believe that few would argue that what we do best now is the analysis of straight-line key distribution and authentication protocols, in which the lowest level of abstraction used is a black-box model of a cryptographic algorithm. For these types of protocols there now exist belief logic tools that can do provide a totally automated analysis [10]. On a somewhat deeper level there are a number of state-based analysis tools that can do a more thorough analysis with minimal input from the user. High-level languages like CAPSL [57] also make it easy to specify these protocols in a way usable by the tools.

In an earlier version of this paper [50] we conjectured that the formal methods tools would work best as animators for "back-of-the-envelope" sketches of protocols so that they could be thoroughly examined before the effort was made to develop them into more finished products. As a matter of fact, something else seems to be happening. The best applied use of formal cryptographic protocol analysis so far appears to be the analysis of standards. Since a protocol must be standardized before it can be of practical use, and since much protocol design and adaptation is done as part of the standardization process, this appears to be a logical place to apply formal methods to get the maximum results. Moreover, since standards are usually publically available, they are generally the first things protocol analysts turn to when they want to show that they can apply their techniques to practical systems. Thus we have seen analyses of protocols such as SSL/TLS [59], [62], Kerberos [7], [13], SET [6], and IKE [47], as well as Bolignano's analysis of C-SET and some subsets of SET (see [9] for a discussion of some the techniques used for this last). The trick now, of course, is to integrate the formal analysis into the standardization process. With the exception of Bolignano's work, the analyses cited above were done after the standardization of the protocol in question had progressed to the point where it was no longer easy to change, and, in many cases, well after the process was complete. But, although the ability to find flaws in published standards is useful (especially for anybody who wants to consider using them), the finding of such flaws will not lead to changes in the standard itself unless it is done early and is carefully integrated into the design process.

Recently we have been trying to put these principals in action by performing an analysis of the Group Domain of Interpretation Protocol for the Internet Engineering Task Force. While

existing tools are apparently not yet mature enough to have the protocol designers themselves do the analysis, we did find that it was possible to work closely with them, giving them feedback as the protocol design progressed. This not only helped us in verifying that our formal specification correctly captured what the designers had in mind, but that many of the problems we found in the protocol specification and in the requirements specification could be fixed before we proceeded to the actual analysis. Indeed, we found that most of the problems we found were identified while writing the specification, although the most subtle and elusive were found in the actual analysis.

We also found that the benefits were two-fold; not only were we able to help in the identification or fixing of bugs, but the fact that a formal analysis had been performed increased confidence in the soundness of the protocol, and thus appears to helping speed up its acceptance by the IETF. We hope that this will spark interest in others doing similar integrations of analysis with protocol development.

V. Conclusion

In this paper we have given a brief outline of the state of the art of cryptographic protocols and shown many directions in which it could be extended. Most conclusions to papers include suggestions for further research. Since this paper consists of nothing but suggestions for further research, we will forgo that here. However, we do note that we did not intend our list of topics to be exhaustive, and we imagine that others who look at this area will come up with other topics as well. Indeed, we imagine that there will be many other areas that come to light as research progresses.

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*, 5(2):103–127, Spring 2002 .

[3] J. Alves-Foss. Multiprotocol attacks and the public key infrastructure. In *Proc. 21st National Information Systems Security Conference*, pages 566–576, Arlington, VA, October 1998.

[4] J. Alves-Foss. Provably insecure mutual authentication protocols: The two party symmetric encryption case. In *Proc. 22nd National Information Systems Security Conference.*, Arlington, VA, 1999.

[5] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *CONCUR00*, number 1877 in LNCS. Springer, 2000.

[6] G. Bella, F. Massacci, L. C. Paulson, and P. Tramontano. Formal verification of cardholder registration in SET. In F. Cuppens, editor, *Computer Security – ESORICS 2000*, pages 159–174. Springer LNCS 1895, 2000.

[7] G. Bella and L.C. Paulson. Kerberos version iv: inductive analysis of the secrecy goal. In Jean-Jacques Quisquater et al., editor, *Computer Security – ESORICS 98*, pages 361–375. Springer LNCS 1485, 1998.

[8] J. Benaloh, B. Lampson, D. Simon, T. Spies, and B. Yee. The private communication technology protocol. draft-benaloh-pct-00.txt, October 1995.

[9] D. Bolignano. Towards the formal verification of electronic commerce protocols. In *Proceedings of the 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

[10] S. Brackin. Evaluating and improving protocol analysis by automatic proof. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1998.

[11] J. Bryans and S. Schneider. CBS, PVS, and a recursive authentication protocol. In *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 3-5 1997. available at http://dimacs.rutgers.edu/Workshops/Security/.

[12] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, February 1990.

[13] F. Butler, I. Cervesato, A. Jaggard, and A. Scedrov. A formal analysis of some properties of Kerberos V using MSR. In *Proceedings of the 15th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2002.

[14] 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.

[15] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. A meta-notation for protocol analysis. In *Proceedings of 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[16] R. Chadha, M.I. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *8th ACM Conference on Computer and Communications Security*, pages 176–185. ACM, November 2001.

[17] D. Chaum. Untraceable electronic mail, return addresses and digital signatures. *Communications of the ACM*, 24(2):84–88, February 1981.

[18] E. Clark, S. Jha, and W. Marrero. Partial order reductions for security protocol verification. In *Tools and Algorithms for the Construction and Analysis of Systems*, 2000.

[19] E. Cohen. TAPS:a first-order verifier for cryptographic protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 144–158. IEEE Computer Society Press, June 2000.

[20] Z. Dang and R. A. Kemmerer. Using the ASTRAL model checker for cryptographic protocol analysis. In *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997. available at http://dimacs.rutgers.edu/Workshops/Security/.

[21] G. Denker, J. Millen, and H. Ruess. The CAPSL integrated protocol environment. Technical Report SRI-CSL-2000-02, SRI International, 2000.

[22] D. Dolev, S. Even, and R. Karp. On the Security of Ping-Pong Protocols. *Information and Control*, pages 57–68, 1982.

[23] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[24] A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 203–212. IEEE Computer Society Press, June 1999.

[25] B. Dutertre and S. Schneider. Using a PVS embedding of CSP to verify authentication protocols. In *TPHOLS'97*, 1997.

[26] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Proceedings of the 24th IEEE Symposium on the Foundations of Computer Science*, pages 34–39. IEEE Computer Society Press, 1983.

[27] N. Ferguson and B. Schneier. A security evaluation of IPSec. In M. Blaze, J. Ioannides, A. Keromytis, and J. Smith, editors, *The IPSec Papers*. Addison-Wesley, to appear.

[28] M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *14th IEEE Computer Security Foundations Workshop*, pages 160–173. IEEE Computer Society, 2001.

[29] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2), February 1999.

[30] 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.

[31] 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.

[32] J. Guttman and J. Thayer Fábrega. Protocol independence through disjoint encryption. In *Proceedings of the 13th Computer Security Foundations Workshop*, pages 24–34. IEEE Computer Society Press, July 3-5 2000.

[33] D. Harkins and D. Carrel. The internet key exchange (IKE). RFC 2409, Internet Engineering Task FOrce, November 1998. available at `http://ietf.org/rfc/rfc2409.txt`.

[34] J. Heather and S. Schneider. Towards automatic verification of authentication protocols on an unbounded network. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2000.

[35] N. Heintze and J. D. Tygar. A model for secure protocols and their composition. *IEEE Transactions on Software Engineering*, 22(1):16–30, Jan. 1996.

[36] A. Huima. Efficient infinite-state analysis of security protocols. presented at FLOC'99 Workshop on Formal Methods and Security Protocols, July 1999.

[37] P. Karn and W. Simpson. Photuris: Session-key management protocol. RFC 2522, Internet Engineering Task Force, March 1999.

[38] J. Kelsey and B. Schneier. Chosen interactions and the chosen proto-

col attack. In *Security Protocols, 5th International Workshop April 1997 Proceedings*, pages 91–104. Springer-Verlag, 1998.

[39] R. Kemmerer. Using formal methods to analyze encryption protocols. *IEEE Journal on Selected Areas in Communication*, 7(4):448–457, 1989.

[40] S. Kremer and J.-F. Raskin. A game-based verification of non-repuditaion and fair exchange protocols. In *CONCUR 2001-Concurrency Theory*. Springer-Verlag LNCS 2154, 2001.

[41] P. Lincoln, J. MItchell, M. Mitchell, and A. Scedrov. Probabilistic polynomial-time equivalence and security analysis. In J. Wing, J. Woodcock, and J. Davies, editors, *FM'99 - Formal Methods*, pages 776–793. Springer-Verlag LLNCS 1709, September 1999.

[42] D. Longley and S. Rigby. An automatic search for security flaws in key management schemes. *Computers and Security*, 11(1):75–90, 1992.

[43] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Software - Concepts and Tools*, 17:93–102, 1996.

[44] G. Lowe. Towards a completeness results for model checking security protocols. *Journal of Computer Security*, 7(2,3):89–146, 1999.

[45] W. Marrero. A model checker for authentication protocols. In *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997. available at http://dimacs.rutgers.edu/Workshops/Security/.

[46] C. Meadows. Applying Formal Methods to the Analysis of a Key Management Protocol. *Journal of Computer Security*, 1:5–53, 1992.

[47] C. Meadows. Analysis of the Internet Key Exchange protocol using the NRL Protocol Analyzer. In *Proceedings of the 1999 Symposium on Security and Privacy*. IEEE Computer Society Press, May 1999.

[48] C. Meadows. A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[49] C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In P. Degano, editor, *Proceedings of the First Workshop on Issues in the Theory of Security - WITS'00*, pages 87–92, Geneva, Switzerland, July 8-9 2000.

[50] C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DISCEX 2000*, pages 237–250. IEEE Computer Society Press, January 2000.

[51] C. Meadows and P. Narendran. A unification algorithm for the group diffie-hellman protocol. In *Proceedings of WITS'02*, January 2002.

[52] 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.

[53] M. J. Merritt. Cryptographic Protocols. Ph.d. thesis, Georgia Institute of Technology, 1983.

[54] J. Millen and H. Ruess. Protocol-independent secrecy. In *2000 IEEE Security and Privacy Symposium*. IEEE Computer Society Press, May 2000.

[55] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *8th ACM Conference on Computer and Communication Security*, pages 166–175. ACM SIGSAC, November 2001.

[56] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: protocol security analysis. *IEEE Transactions on Software Engineering*, SE-13(2), 1987.

[57] Jonathan K. Millen. CAPSL: Common Authentication Protocol Specification Language. Technical Report MP 97B48, The MITRE Corporation, 1997. See http://www.csl.sri.com/ millen/capsl.

[58] J. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using Murϕ. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 141–151. IEEE Computer Society Press, May 1997.

[59] J. C. Mitchell, V. Shmatikov, and U. Stern. Finite-state analysis of SSL 3.0. In *Seventh USENIX Security Symposium*, pages 201–216, San Antonio, 1998.

[60] R. M. Needham and M. D. Schroeder. Using Encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[61] L. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, June 1997.

[62] L. C. Paulson. Inductive analysis of the Internet protocol TLS. *ACM Transactions on Computer and System Security*, 2(3):332–351, 1999.

[63] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[64] D. Pavlovic and D. R. Smith. Guarded transitions in evolving specifications. Technical report, Kestrel Institute, Feb 2002.

[65] O. Pereira and J.-J. Quisquater. A security analysis of the Cliques protocols suites. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 73–81. IEEE Computer Society Press, June 2001.

[66] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the 2000 IEEE Symposium on Security and Prvacy*, pages 247–262. IEEE Computer Society Press, May 2000.

[67] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM TISSEC*, June 1999.

[68] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *8th IEEE Computer Security Foundations Workshop*, pages 98–107. IEEE Computer Society, 1995.

[69] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop*, pages 174–190. IEEE Computer Society, 2001.

[70] S. Schneider and A. Spirodopoulos. CSP and anonymity. In *Proceedings of ESORICS 96*. Springer-Verlag, 1996.

[71] SET Secure Electronic Transactions LLC. *The SET Specification*. available at `http://www.setco.org/set_specifications.html`.

[72] V. Shmatikov. Probabilistic analysis of anonymity. In *Proceedings of the 15th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2002.

[73] V. Shmatikov and J. Mitchell. Finite-state analysis of two contract-signing protocols. *Theoretical Computer Science*, (2):419–450 June 2000. Special Issue on Security, Roberto Gorrieri, ed.

[74] D. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1):47–74, 2001.

[75] S. Stoller. A bound on attacks on authentication protocols. presented at 1999 Workshop on Formal Methods and Security Protocols, available at http://www.cs.indiana.edu/hyplan/stoller/, July 1999.

[76] S. Stubblebine and V. Gligor. On message integrity in cryptographic protocols. In *Proceedings of the 1992 Symposium on Security and Privacy*, pages 85–104. IEEE Computer Society Press, May 1992.

[77] S. Stubblebine and C. Meadows. Formal characterization and automated analysis of known-pair and chosen-text attacks. *IEEE Journal on Selected Areas in Communication*, 18 (4): 571-581, April 2000.

[78] P. Syverson and S. Stubblebine. Group principals and the formalization of anonymity. In J. Wing, J. Woodcock, and J. Davies, editors, *FM'99 - Formal Methods*, pages 814–833. Springer-Verlag LLNCS 1708, 1999.

[79] P. F. Syverson and P. C. van Oorschot. On unifying some cryptographic protocol logics. In *1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 14–28. IEEE Computer Society, May 1994.

[80] F. Thayer Fábrega, J. Herzog, and J. 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.

[81] F. Thayer Fábegra, J. Herzog, and J. Guttman. Mixed strand spaces. In *Proceedings of the IEEE 1999 Computer Security Foundations Workshop*, pages 72–82. IEEE Computer Society Press, June 1999.

[82] B. Timmerman. Secure adaptive dynamic traffic masking. In *Proceedings of the 1999 New Security Paradigms Workshop*. ACM, pages 13–24. ACM, 2000.

[83] P. C. van Oorschot. Extending cryptographic logics of belief to key agreement protocols (extended abstract). In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 232–243. ACM, November 1993.

[84] B. Venkatraman and R. Newman-Wolfe. Capacity estimation and auditability of network covert channels. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*. IEEE Symposium on Security and Privacy, May 1995.

[85] M. Wright, M. Adler, B. N. Levine, and C. Shields. An analysis of the degredation of anonymous protocols. In *Proceedings of the ISOC Network and Distributed System Security Symposium*, February 2002.

[86] J. Zhou. Fixing a security flaw in IKE protocols. *Electronics Letters*, 35(13):1072–1073, June 1999.