Open Issues in Formal Methods for Cryptographic Protocol Analysis

Catherine Meadows
Code 5543
Naval Research Laboratory
Washington, DC 20375
meadows@itd.nrl.navy.mil

Abstract

The history of the application of formal methods to cryptographic protocol analysis spans nearly twenty years, and recently has been showing signs of new maturity and consolidation. A number of specialized tools have been developed, and others have effectively demonstrated that existing general-purpose tools can also be applied to these problems with good results. However, with this better understanding of the field comes new problems that strain against the limits of the existing tools. In this paper we will outline some of these new problem areas, and describe what new research needs to be done to to meet the challenges posed.

1 Introduction

The history of the application of formal methods to cryptographic protocol analysis spans nearly twenty years, and recently has been showing signs of new maturity and consolidation. A number of specialized tools have been developed, and others have effectively demonstrated that existing general-purpose tools can also be applied to these problems with good results.

However, with this better understanding of the field comes new problems that strain against the limits of the existing tools. Tools as they exist today are mainly intended to be applied to the problem of showing that a key has been correctly authenticated. This was the usual application for cryptographic protocols in the past, and remains a common one today. But cryptographic protocols are now being applied to new types of problems, such as group communication, financial transactions, and negotiation of algorithms as well as keys. They also face new types of threats, such as denial of service, and older threats, such as traffic analysis, that are becoming more prominent. In many cases the existing tools are not capable of dealing with these problems. However we believe that many of the existing techniques could be adapted so they could be applied successfully. In this paper we will outline some of these new areas, and describe what new research needs to be done to to meet the challenges posed.

Many of the ideas on new directions that are described in this paper were developed during the application of the NRL Protocol Analyzer to the analysis of the Internet Key Exchange protocol and during the development of a set of formal requirements for the Secure Electronic Transactions Protocols, both of which were done as part of the DARPA project Formal Analysis of Internet Security Protocols. One of the purposes of this project was to see how far current tools could be pushed to analyze complex protocols that must satisfy new types of requirements, and also to find out where our tools need to be improved. In this paper we show how this research contributed to our understanding of the areas discussed in this paper.

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. In Section Three we describe the NRL Protocol Analyzer and the IKE and SET protocols that we examined, describing the features that made them a challenge. In Section Four we describe seven different emerging areas: open-ended protocols, denial of service, anonymous communication, high fidelity, composability, negotiation of complex data structures, and getting the products of our research into the real world. Where appropriate, we refer back to our work on IKE and SET for motivation. Section Five concludes the paper.

2 The History and Current State of Formal Cryptographic Protocol Analysis Tools

Cryptographic protocols are protocols that use cryptography to distribute keys and authenticate principals and data over a network. The network is 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. Because of this, such protocols are often subject to nonintuitive attacks which are not easily apparent even to a careful inspector, especially when assumptions about the environment in which the protocol operates change. For example, as is pointed out in [39], the Needham-Schroeder public key protocol [37], which was intended to be used for communication between parties that trust each other, and is secure as long as those assumptions hold, become subject to a man-in-the-middle attack if we assume that one of the communicating parties may be dishonest [24], an assumption which has become more likely in a web-based environment.

For these reasons, it has long been realized that formal methods can be a useful method for analyzing 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 [37]. However, the first work that was actually done in this area was done by Dolev and Yao [10], and slightly later by Dolev, Even, and Karp [9], 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 [13], 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 [34], the NRL Protocol Analyzer [27], and the the Longley-Rigby tool [23]. Others applied general-purpose formal methods to the problem [20]. 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 [27], and the use of the Longley-Rigby tool to find a flaw in a banking protocol [23].

However, this still remained a fairly esoteric area until the publication of the Burrows, Abadi, and Needham logic [5] 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 knowledge and belief, which consists of 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 receives 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 the 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 [3].

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 [24]. Work since then has progressed in applying both model checkers [36, 7] and theorem provers [41, 12] to the problem, as well as in the design of special-purpose model checkers [19, 43, 25] and the use of specialized tools originally intended for somewhat different applications [11].

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 [35], the Common Authentication Protocol Language, which is intended to provide a common specification language for cryptographic protocol analysis tools. Even more recently, Thayer, Herzog, and Guttman [14] have 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 [43] and as a framework in which to express theoretical results [44]. This trend has promising implications for the integration of future tools and the incorporation of new theoretical results into these tools.

To sum up, at present we are rapidly approaching, if we have not already reached, a state in which we will have a number of different tools available that will be able to verify safety properties such as authentication and secrecy by performing a state space 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. This is not everything; such tools will not catch errors that arise from implementation details that go beyond the journal-level specification, the Dolev-Yao model leaves out some important attacker capabilities such as cryptanalysis, and, 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 in undecidable, [13, 18, 6], 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.

3 IKE, SET, and the NRL Protocol Analyzer

3.1 Overview

In this section we describe the IKE and SET protocols that we analyzed for the DARPA project Formal Analysis of Internet Security Protocols, as well as the tool we applied, the NRL Protocol Analyzer. We concentrate, not on the analyses themselves, which are described elsewhere, but on the challenges that each posed. We identify both those challenges that our tools proved capable of meeting, and those areas where we felt our tools fell short.

3.2 The NRL Protocol Analyzer

The NRL Protocol Analyzer is a formal methods tool for analyzing security properties of cryptographic protocols. It uses automatic invariant generation to limit a potentially infinite search space in combination with exploration of the remaining space to generate attacks on insecure protocols and provide security proofs for secure ones, even in the face of a potentially unlimited number of protocol executions or an unlimited number of intruder actions.

Protocols in the Analyzer model are specified as communicating state machines, one of which is a hostile intruder following the Dolev-Yao model who can read all traffic, modify or delete traffic, perform cryptographic operations, and may be in cooperation with some legitimate users of the system.

The user of the Analyzer attempts to determine whether or not a protocol is secure by specifying an insecure state. This state can be specified not only in terms of values of local state variables and terms known by the intruder, but sequence of events that should or should not have occurred. For example, the user could ask the Analyzer to look for a state in which the same key has been accepted twice by a principal (two events occurring) or a state in which a responder B accepts a key as good for communicating by an initiator A, but in which A never initiated the protocol (one event having occurred and another event not having occurred previously). The Analyzer works backwards from that state until it has explored the search space exhaustively, so that each path produced either begins in an initial state (describing an attack) or an unreachable state. Thus, like the other tools we mentioned earlier, it can be used fo prove safety properties such that no party is authenticated incorrectly, but not liveness properties such that an authentication always completes.

The Analyzer makes no assumptions about limits on the number of protocol executions, the number of principals performing the different executions, the number of interleaved executions, or the number of times cryptographic functions are applied. This results in a search space that is originally infinite. However, the Analyzer provides means for specifying and proving inductive lemmas about the unreachability of infinite classes of states. This allows the user to narrow down the search space so that in many cases an exhaustive search is possible.

We also made use of the NRL Protocol Analyzer Temporal Requirements Language (NPATRL) [48]. NPATRL is a temporal logic language allowing us to specify desirable protocol properties in terms of desirable or undesirable sequence of events. An NPATRL requirement is applied to the NRL Protocol Analyzer by taking the negation of the requirement and using that as an insecure state for the Analyzer to prove unreachable.

The Analyzer has been applied to a number of different cryptographic protocols, and has found flaws in several. In some cases the flaws had not been discovered before. Examples of protocols the Analyzer has been used to examine are the Simmons Selective Broadcast Protocol [27], the Burns-Mitchell Ticket Granting Protocol [32], and an early version of the Encapsulating Security Protocol [46]. The Analyzer has also been used to prove security properties of a number of other protocols, by performing an exhaustive search of the finite space that is left after the necessary lemmas have been proved (see [28, 29]. A more detailed description of the Analyzer is given in [33].

The NRL Protocol Analyzer, as we see, operates within the same paradigm as most specialized tools for cryptographic protocol analysis, and indeed, was one of the first tools to make use of it. That is, it is based on the Dolev-Yao model and it concentrates mainly on proving authentication properties. Thus it is well suited for testing the limits of that paradigm.

3.3 IKE

The Internet Key Exchange protocol (IKE) is a key exchange protocol being developed by the IP Security Protocol (IPSEC) Working Group of the Internet Engineering Task Force (IETF). It is intended to provide the security support for client protocols of the Internet Protocol. As such, it does much more than simply distribute keys; it also is intended to be used to establish *Security Associations* that specify such things as the protocol format used, the cryptographic and hashing algorithms used, and other necessary features for secure communication. Since it is intended to be flexible, it supports a number of different types of key exchange options, including digital signatures, public key encryption, and conventional encryption using shared keys. The Diffie-Hellman algorithm is used to generate shared key material, but is optional in some cases. IKE has evolved from a number of different protocols, including ISAKMP [26], Oakley [38], the Station-to-Station protocol [8], and SKEME [21], the last two of which influenced the development of Oakley.

A typical key establishment protocol proceeds in one phase, in which two parties use master keys to establish shared keying material. IKE, however, proceeds in two such phases. In the first phase, two entities use master keys to agree, not only on keying material, but on the various mechanisms (e.g. cryptographic algorithms, hash functions, etc.), that they will use in the second phase. The keying material and set of mechanisms thus agreed upon is called a *security association.* In the first phase protocol, the initiator proposes a number of possible security associations to the responder, who picks one. In the second phase, the keys and mechanisms produced in the first phase are used to agree upon new keys and mechanisms that will be used to protect and authenticate further communications. The security association established in Phase One is bidirectional, so the initiator in the first phase can be either initiator or responder in the second phase.

At the time we analyzed IKE, it could be used in four different modes: main mode, aggressive mode, quick mode, and new group mode. Main and aggressive modes are used in Phase One negotiations, quick mode is used in Phase Two negotiations, and New Group Mode is used to change the Diffie-Hellman group. Both main and aggressive mode can be implemented using several different types of authentication; there is a different protocol for each one.

The main goals of IKE are the authentication of security associations, and the authentication and generation of keys. However, IKE has some other secondary goals, as well. A good deal of thought has gone into making IKE resistant to denial of service attacks by preventing it from requiring principals to engage in resource-intensive activities until a "reasonable" amount of authentication has occurred. IKE is also designed to be somewhat resistant to traffic analysis, in that in Main Mode identities are encrypted.

In our analysis of IKE, described in detail in [29], we concentrated on verifying that keys were properly authenticated, and that portions of the Security Associations were authenticated. We did not attempt to verify to what degree the protocol resistant to traffic analysis or denial of service; this was beyond the scope of our tool. However, it is clear that such an analysis would have been useful. These issues are discussed in more detail in Section 4.2 on Denial of Service and Section 4.3 on Anonymous Communication.

One of the main challenges in analyzing IKE was the sheer proliferation of related subprotocols. Since we wanted to verify, among other things, that one protocol could not be confused with another, we analyzed as many subprotocols together as we possibly could. The NRL Protocol Analyzer proved adequate to this task, but not before undergoing a major overhaul designed to increase its robustness, and some serious thinking about how to identify possible interactions. These issues are discussed in depth in Section 4.5 on Composability.

3.4 SET

The SET Protocol [31] is a protocol sponsored by major credit card companies and others that is intended to provide a standard for safe, secure credit card transactions over the Internet. ('SET' stands for 'Secure Electronic Transactions'.) As such, it is intended to supply an electronic version of the paper system that exists today. However, there are a number of risks connected with use of the Internet that do not arise in the paper world, or at least are not considered as severe. These arise from the difficulty of identifying participants in transactions and the difficulty of ensuring the private information sent over the Internet remains so. SET is intended to reduce these risks by introducing cryptographic means to protect sensitive information such as credit card numbers and to provide authentication of parties involved in a credit card transaction.

A payment transaction in the SET protocol involves three parties: a customer, a merchant, and an application payment gateway. The customer presents a purchase request to the merchant, which includes credit card information and a proposed purchase amount. The purchase request is identified with a transaction ID. The merchant then passes the request along to the gateway, together with a request that a certain amount (not necessarily equal to the purchase amount) be authorized. The gateway then checks the customer's credit, authorizes a certain amount, and passes this information back to the merchant. The merchant passes this information back to the customer. Either at the same time as the authorization request, or later, the merchant presents a capture request to the gateway for the same transaction, requesting that a certain amount of money be captured. The gateway approves a certain amount which may or may not be equal to the amount requested. The merchant then passes this information back to the customer.

The authentication structure of the SET protocol is complex. Messages between merchant and gateway are always authenticated using digital signatures, as are messages from the merchant to the customer. Digital signatures for authentication for messages from customer to merchant are optional, although they may be made mandatory by a particular application. However, it is in the authentication of forwarded messages that the structure really becomes interesting. The message from customer to merchant includes information that is needed by the gateway but may be hidden from the merchant, such as credit card number (PAN, i.e., Primary Account Number) and expiration date. Also included, when customer digital signatures are used, is a data item called the PANSecret, which is known only by the customer and gateway, but not the merchant. This is not available when customer digital signatures are not used, since it is generated as part of the certificate registration process. This information is protected by the use of a *dual signature*. Two hash functions are computed, one over the the data to be kept hidden from the merchant, and the other over the data to be revealed to it, which includes a hash over the purchase amount and order description that the customer and merchant agreed to offline. The hidden data is encrypted using the gateway's public key. The customer then computes a digital signature (if customer signatures are used) over the two hashes. The signature, the two hashes, and the encrypted and unencrypted information are sent to the merchant. The merchant verifies the signature and forwards the information, including the signature if any, to the gateway. When the gateway receives the message, it verifies the signature, if any, and also verifies the PAN and PANSecret. Whether or not signatures are used, it also verifies the PAN and and the customer's portion of the PANSecret (if any) with the credit card issuer, although this may be done offline.

Authentication of gateway to customer via the merchant is much simpler: there is none. Any information from the gateway that the merchant passes on to the customer is authenticated only by the merchant's signature.

There are also a number of options available. A customer has the option of sending an initialization message prior to its purchase request, which allows it to obtain more up-to-date certificates from the merchant, and allows the merchant to send back a random challenge which it can use to verify the freshness of the customer's subsequent purchase request. When an initialization message is sent, the transaction ID is jointly created by customer and merchant. When no initialization message is sent, the customer may create the transaction ID, or it may be jointly created by the customer and merchant. The gateway also has the option, depending upon the policy followed, of sending the customer's PAN to the merchant (the PANSecret, however, is never sent). There are also protocols for inquiring about the status of an order, cancelling an order, etc.

We have not yet completed our analysis of SET, but we have written a formal specification of the SET requirements using the NPATRL requirements language [31]. This turned out to be a nontrivial task. Although the NPATRL language was expressive enough to state the requirements, their complexity meant that any such specification would be incomprehensible unless we introduced some higher-level framework in which to organize it. We describe the problem and our solution in more detail in Section 4.6 on Negotiation of Complex Data Structures.

4 Emerging Problems

4.1 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, in general each message is a fixed structure that is composed of a bounded number 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 protocol 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; no precise limit is put on them by the protocol specification. The most obvious 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), 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, recently Zhou [54] and independently Ferguson and Schneier [15] 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.

So far, there has been very little work on applying formal analysis techniques to these kinds of problems in cryptographic protocols. One notable exception has been the work of Paulson [40] on the 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 [4] 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. Thus it may be the case that techniques that were developed to deal with the unboundedness that arises first out of the ability of an intruder to perform an arbitrary number of message operations, and secondly out of the possible execution of an unbounded number of protocol operations in parallel, should be applicable to other types of open-ended protocols as well, although they will probably require some adaptation and expansion.

4.2 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 of 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.

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.

We have been working on a framework [30] 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 [17] 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.

4.3 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 [16], the Anonymizer and Crowds [42], 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.

Other systems provide other types of anonymity. Proxymate distributes pseudonyms for dealing with third parties. The Rewebber supports anonymous publishing. Anonymous remailers support anonymous email. Indeed, we can safely assume that, for any application involving communication over the Internet, there are situations in which one might be concerned about preserving anonymity and preventing traffic analysis.

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 [51], 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 [52] 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, whic 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. 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 a very realistic assumption. However, attackers who only reside at single nodes and do not communicate are not very realistic either. The work by Syverson and Stubblebine on group principals [47] 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.

In summary, we believe that the analysis of anonymity protocols pose a 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.

4.4 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 cryptosystem 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. 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 [45] and reproducing known attacks [46]. From an entirely different angle, work has also been ongoing on introducing some of the polynomial-time reduction techniques used by cryptographers into the framework used by formal methods, making it possible to reason more precisely about the interaction of a protocol with the cryptographic algorithms that is uses [22] Finally, recently new interest has been shown in revisiting the problems of secrecy models in cryptographic protocols, thus going beyond the standard black-box assumptions [53]. Thus we see new interest in the problem of high fidelity from a number of different sides.

4.5 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 [1] 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.

Early work on composability [18, 17] concentrated on determining under what conditions protocols could be guaranteed to be composable. The early results led to rather stringent requirements: in essence, they required the fail-stop property [17] or something very similar to it [18]. Thus they did not have much practical application.

More recent work has concentrated, not on designing protocols that are guaranteed to be composable, 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, thus limiting the state explosion problem. 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 that must process a large number of rules), but we were suprised at the dramatic speedup it caused, (at least threefold for the IKE analysis [29]). 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 [49]. 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. The techniques used for choosing the set of terms and for using them in the proof are specific to the protocols used in [14], but it is likely that they could be generalized into heuristics that could be applied more widely. We think that it is likely that these techniques will continue to be useful as they are further refined.

We also believe that the last word has not been written on designing protocols that are guaranteed to be composable, or are at least easier to prove composable. Recently some promising work has been done on designing protocols that are verifiable by model-checkers [24, 44]. A model-checker checks that a protocol is secure by simulating its interaction with an intruder. Since simulation is used, only a finite number of executions of the protocol can be examined. Thus it is helpful to know if there is some finite number of executions such that the examination of this finite number is enough to guarantee security. In general, the answer to this is no [13, 18, 6], but it is still possible, as is shown by the work of Lowe [24] and Stoller [44], to put design constraints on protocols that guarantee them to be verifiable through examination of a finite number (and manageable) number of executions. Moreover, the constraints involved are much more realistic than the fail-stop constraint used in the early composability results.

We believe that this work could be used in the analysis of composability in the following way. Consider a protocol that integrates a set of straight-line subprotocols (where a "straight-line" protocol is defined to be one that involves no if-then-elses or loops) by proceeding in two stages. In the first stage, an initiator chooses which of a suite of straight-line protocols it will execute. In the next stage it executes the subprotocol. In order for two subprotocols to interact harmfully, at least two executions of the protocol must have taken place. If we can modify Lowe's and/or Stoller's results so that they would still hold in such an environment, then they could be used to limit the complexity of checking for an interaction by limiting the size of the set of subprotocols that needs to be checked at any one time.

4.6 Negotiation of Complex Data Structures

Key distribution and agreement protocols usually negotiate agreement upon a single field: the key. Most work on formal analysis of cryptographic protocols has concentrated on this problem. But once more structure is introduced into the data to be agreed upon, new problems of maintaining the consistency of the data emerge. For example, Thayer, Herzog, and Guttman [50] recently discovered a new attack on the Otway-Rees protocol, which distributes a key and a key identifier together. In this attack the principals wound up agreeing on the same key identifier but not the same key. Although the Otway-Rees protocol had been carefully studied in the past, this possible behavior had not been uncovered before.

Many of the newer protocols negotiate agreement on much more complex structures than this. As we saw earlier, the Internet Key Exchange Protocol negotiates agreement on a complex data structure that not only includes multiple fields, but is also open ended. SET goes IKE one further by having principals agree on a data structure the contents of whose fields are generated by different members of a three-party protocol as the protocol progresses, so that we are dealing with a data structure that evolves as the protocol completes. Moreover, different parts of the structure may be kept secret from different principals, so it is possible that no one principal knows the contents of all the fields in the data structure, even after the protocol completes.

In spite of this complexity, we found the tool we were using, the NRL Protocol Analyzer, adequate to this task, except for the case of open-ended data structure, described in Section 4.1. The hard part was understanding and specifying the protocol requirements that needed to be satisfied, especially for SET. We were fortunate to have the NPATRL requirements language [48], that had been developed specifically to specify complex protocol requirements for the NRL Protocol Analyzer. However, we still found it necessary to augment it in order to reason about agreement on data structures with possibly unknown or not yet defined fields. We did this in terms of specifying requirements on agreement on projections of the entire data structure. This was not difficult to do, but did require an enhancement to the NPATRL language.

We can see other types of data structures emerging as well. Certificate hierarchies are one example. Since any protocol that manages certificate hierarchies must deal with the revocation of certificates as well, this brings up the problem of introducing negation. There is also a growing body of work on using certificates to implement security policies (e.g. Policymaker [2]). Although there has been a good deal of research in the development of policy definition languages, little of this work has concentrated on examining their interaction with the protocols that implement them. It is still early to tell if there are any problems here the require special attention, but the area bears watching.

4.7 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 not to concentrate 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.

I think 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 cryptosystem. For these types of protocols there now exist belief logic tools that can do provide a totally automated analysis [3]. 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 [35] also make it easy to specify these protocols in a way usable by the tools.

We have noted, of course, that the reality is often much more complex than the simple protocols that these tools were devised to analyze. But most of the complex protocols had their origins in the simpler journal level protocols; for example the complex Internet Key Exchange protocol is in part derived from the very simple Station to Station protocol. Nor are the "simple" protocols as transparent as they might appear at first glance. Lowe's analysis of the Needham-Schroeder public key protocol and Thayer, Herzog, and Guttman's analysis of the Otway-Rees protocol show that it is possible for possibly harmful properties to go undiscovered for years. Thus, it is important to have the ability to recognize and fix potential problems up front. If we think of the existing tools as providing the potential for animating "back-of-the-envelope" sketches of protocols so that they can be thoroughly examined before the effort is made to developed them into more finished products, I think we will have an idea of how our tools as they exist now can be made immediately useful to protocol designers.

5 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 exclusive. Indeed, we imagine that there will be many other areas that come to light as research progresses.

6 Acknowledgments

I would like to thank the participants in the FLOC99 Workshop on Formal Methods and Security Protocols for helpful discussions on these topics. This work was sponsored by NSA and DARPA.

References

[1] Josh Benaloh, Butler Lampson, Daniel Simon, Terence Spies, and Bennet Yee. The private communication technology protocol. draft-benaloh-pct-00.txt, October 1995.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Managing trust in information systems. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Socity Press, May 1996.

[3] 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.

[4] 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/.

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

[6] 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.

[7] 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/.

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

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

[11] 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.

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

[13] 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.

[14] F. Javier Thayer Fabrega, 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.

[15] 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.

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

[17] 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.

[18] 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.

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

[20] Richard Kemmerer. Using Formal Methods to Analyze Encryption Protocols. *IEEE Journal on Selected Areas in Communication*, 7(4):448–457, 1989.

[21] H. Krawczyk. SKEME: A versatile secure key exchange mechanism for Internet. In *ISOC Secure Networks and Distributed Systems Symposium*, San Diego, 1996.

[22] 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.

[23] D. Longley and S. Rigby. An Automatic Search for Security Flaws in Key Management Schemes. *Computers and Security*, 11(1):75–90, 1992.

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

[25] 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/.

[26] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet security association and key management protocol (ISAKMP). Request for Comments 2408, Network Working

Group, November 1998. Available at http://www.ietf.org/html.charters/ipsec-charter.html.

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

[28] C. Meadows. Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. In *Computer Security - ESORICS 96*, pages 355–364. Springer-Verlag, September 1996.

[29] 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.

[30] 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.

[31] 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.

[32] Catherine Meadows. A system for the specification and verification of key management protocols. In *Proceedings of the 1991 IEEE Computer Society Symposium in Research in Security and Privacy*. IEEE Computer Society Press, May 1991.

[33] Catherine Meadows. The NRL Protocol Analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[34] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol Security Analysis. *IEEE Transactions on Software Engineering*, SE-13(2), 1987.

[35] 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.

[36] 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.

[37] 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.

[38] H. Orman. The OAKLEY key determination protocol. Request for Comments 2412, Network Working Group, November 1998. Available at http://www.ietf.org/html.charters/ipsec-charter.html.

[39] Susan Pancho. Paradigm shifts in protocol analysis. In *Proceedings of New Security Paradigms Workshop 1999*. ACM, to appear.

[40] 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.

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

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

[43] D. Song. Athena: An automatic checker for security protocol analysis. In *Proceedings of 12th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1999.

[44] 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.

[45] 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.

[46] S. Stubblebine and C. Meadows. Formal characterization and automated analysis of known-pair and chosen-text attacks. *IEEE Journal on Selected Areas in Communication*, to appear.

[47] 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.

[48] 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.

[49] F. Thayer, 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.

[50] J. Thayer, J. Herzong, and J. Guttman. Honest ideals on strand spaces. In *Proceedings of 11th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1998.

[51] B. Timmerman. Secure adaptive dynamic traffic masking. In *Proceedings of the 1999 New Security Paradigms Workshop*. ACM, to appear.

[52] 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.

[53] D. Volpano. Formalization and proof of secrecy properties. In *Proceedings of the IEEE 1999 Computer Security Foundations Workshop*, pages 92–95. IEEE Computer Society Press, June 1999.

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