Formal Verification of Cryptographic Protocols: A Survey

Catherine A. Meadows

Center for High Assurance Computer Systems
Naval Research Laboratory
Washington DC, 20375

Abstract. In this paper we give a survey of the state of the art in the application of formal methods to the analysis of cryptographic protocols. We attempt to outline some of the major threads of research in this area, and also to document some emerging trends. ...

1 Introduction

A cryptographic protocol is meant to provide secure services. However, if the protocol is not designed correctly, it may fail to do so. A hostile intruder may be able to subvert the goals of the protocol by feeding false messages to honest users of the system. If the protocol is not designed to check these false messages adequately, then the intruder's action may result in some security failure such as key compromise or false authentication. Such security flaws in a protocol can be subtle and hard to find; a number of examples exist in the literature of flaws that were not found for some time in protocols that had received extensive hand analysis. Examples include the Needham-Schroeder key distribution protocol [30], which was found by Denning and Sacco [10] to allow an intruder to pass off an old, compromised session key as a new one, the software protection scheme of Purdy, Simmons, and Studier [33], for which Simmons [35] showed how an intruder could combine previously generated messages in such a way that the system could be induced to grant unauthorized access to software, and a protocol in the CCITT X.509 draft standard [9], for which Burrows, Abadi, and Needham [6] showed that an intruder could cause an old session key to be accepted as a new one, whether or not it had been compromised. These examples describe only a few of the documented cases; numerous others exist.

These kinds of problems appear to be well suited for the application of formal methods. They are well-contained enough so that modeling and analysis should be tractable; on the other hand, they are complex enough and the flaws are counterintuitive enough so that an informal analysis may be too prone to error to be reliable. Formal methods have long been used in the analysis of communication protocols in general, and some promising work was done in the analysis of cryptographic protocols in the late 70's and early 80's [13, 12, 27]. But in general, interest in the application of formal methods to cryptographic protocols did not become widespread until the early 90's, when several researchers were able to find heretofore undiscovered security flaws in cryptographic protocols by using formal analysis techniques.

In this paper we give a survey of the state of the art in the application of formal methods to the analysis of cryptographic protocols. In general, we will avoid the discussion of methods, such as zero-knowledge and polynomial reduction, that rely on studying the complexity-theoretic properties of the cryptographic algorithms involved, and which are already well-documented in the literature (e.g. in [15]), and instead concentrate on recently developed methods devised to study properties of protocols that are for the most part independent of the properties of the cryptoalgorithms involved. We will attempt to outline some of the major threads of research in this area, and also to document some emerging trends.

The rest of this paper is organized as follows. In Sections 2 and 3 we describe the two most commonly followed approaches to the applications of formal methods to cryptographic protocol analysis: the use of methods based on communicating state machine models, and the use of logics of knowledge and belief. In Section 4 we will discuss an approach that has not been followed by as many people, but has been successful in modeling some subtle properties of cryptographic protocols, that is, the use of algebras to model the state of an individual's knowledge about words used in a protocol. In the remaining sections we will discuss some open issues and emerging trends in the formal analysis of cryptographic protocols. These include model granularity, requirements modeling, and the use of formal methods in the design of new protocols as opposed to the analysis of existing ones.

2 Methods Based on State Machines

Most versions of the state-machine approach embody at least some aspects of the work of Dolev and Yao [13] and of Dolev, Even, and Karp [12]. In the Dolev-Yao model, the network is assumed to be under the control of a intruder who can read all traffic, alter and destroy messages, create messages, and perform any operation, such as encryption, that is available to legitimate users of the system. However, it is assumed, at least initially, that the intruder does not know any information that is to be kept secret, such as encryption keys belonging to honest users of the system.

Since the intruder can prevent any message from reaching its destination, and since he can also create messages of his own, we may treat any message sent by an honest user as a message sent to the intruder and any message received by an honest user as a message received from the intruder. Thus the system becomes a machine used by the intruder to generate words. These words obey certain rewrite rules, such as the fact that encryption and decryption with the same key cancel each other out. Thus we can think of the intruder as manipulating a term-rewriting system. If the goal of the intruder is to find out a word that is meant to be secret, then the problem of proving a protocol secure becomes a word problem in a term-rewriting system. Dolev et al. use this observation to develop several algorithms to analyze restricted classes of protocols in terms of their properties as term-rewriting systems.

The Dolev-Yao model is too restricted to be useful for the analysis of most protocols. First, it can only be used to detect failures of secrecy; second, it does not allow participants to remember state information from one state to the next. Thus, most protocol analysis methods that use the intruder-based Dolev-Yao model as a basis generally augment it with more conventional protocol modelling techniques to describe the behavior of the protocol participants.

One of the earliest systems to use a Dolev-Yao approach is the Interrogator developed by Millen [28, 20]. The Interrogator is a software tool that attempts to locate protocol security flaws by an exhaustive search of the state space. In the Interrogator, protocol participants are modeled as communicating state machines whose messages to each other are intercepted by an intruder who can either destroy messages, modify them, or let them pass through unmodified. Given a final state in which the intruder knows some word which should be secret, the Interrogator will try all possible ways of constructing a path by which that state can be reached. If it finds such a path, then it has identified a security flaw. The Interrogator has not yet found a previously unknown attack on a cryptographic protocol, but it has been able to reproduce a number of known attacks [20].

Others have used approaches similar to that of the Interrogator, but have relied upon human intervention to assist in the search. For example, a search tool developed by Longley and Rigby [22] has been used to find a subtle and previously unknown flaw in a hierarchical key management scheme. The chief difference between the Longley-Rigby tool and the Interrogator is that the Longley-Rigby tool allows human intervention. Whenever the system judges that a word cannot be found by the intruder, the user can intervene and determine whether or not that is likely to be the case. If the word is judged to be accessible, this information can be inserted into the database and the search can proceed.

In a different vein, Kemmerer has shown how cryptographic protocols can be modeled in a conventional formal specification language by modeling protocols in Ina Jo [19, 20]. He has also demonstrated how attacks on protocols can be modeled in such a language, and has used a specification animation to "walk through" several such attacks. Like Millen, Kemmerer models cryptographic protocols as communicating state machines. However, because the protocols are modeled in a specification language that has a theorem prover attached to it, it is also possible to use the prover to prove theorems about the security of the specified protocols, by defining security properties as state invariants and proving that these invariants are preserved by each transition, although this has not been yet attempted to any great extent.

The NRL Protocol Analyzer [20] is also based on the Dolev-Yao model, and uses a strategy similar to the Interrogator and the Longley-Rigby tool. As in the case of Millen's Interrogator, one uses the tool to find protocol security flaws by specifying an insecure state and attempting to construct a path to that state from an initial state. Unlike Millen's Interrogator, an unlimited number of protocol rounds are allowed in a single path, so that the state space is infinite. This allows the Analyzer to discover attacks that rely on the intruder's ability to weave several different runs of a protocol together. For example, such an attack was found in [41]. Also unlike the Interrogator, the emphasis is, not only on finding paths to insecure states, but on proving that these states are unreachable. This is made possible by having the user prove that certain paths leading backwards from the insecure state go into infinite loops, never reaching an initial state. Once these paths have been eliminated, the resulting search space is often small enough to search exhaustively. The proofs that paths lead into infinite loops are largely guided by the user; thus the search is much less automated than in the Interrogator.

Although the NRL Protocol Analyzer primarily emphasizes proofs of state unreachability, it can also be used to find flaws in protocols by generating paths to insecure states, and it has been used to find several previously undiscovered security flaws in cryptographic protocols. It has been used [25] to find an authentication flaw in Simmons' Selective Broadcast Protocol [35] and has also been used [24] to find a flaw in Burns and Mitchell's Resource Sharing Protocol [5]. The Analyzer has also been used to demonstrate several flaws that were already known to exist, including one in the Tatebayashi-Matsuzaki-Newman protocol whose flaw is demonstrated in [20], and one in a draft ISO authentication protocol, whose flaw is discussed in [11].

3 Systems Based on Modal Logic

The other approach that has been followed in the application of formal methods to cryptographic protocol analysis is to use modal logics similar to those that have been developed for the analysis of the evolution of knowledge and belief in distributed systems. Such a logic consists of various statements about belief in or knowledge about messages in a distributed system, and inference rules for deriving beliefs from other beliefs andor knowledge from other knowledge and beliefs. A discussion of research in this area is given by Syverson in [42].

Perhaps the best known and most influential such logic was that developed by Burrows, Abadi, and Needham [7], commonly known as BAN logic. BAN logic builds upon statements about messages sent and received throughout the course of a protocol. For example, one such belief, stated informally, would be: "If I believe I've received a message encrypted with key K, and I believe that only Alice and I know K, then I believe that the message was originated by either Alice or me." In an analysis of a protocol, an initial set of beliefs is assumed. Each message received is then mapped to another set of beliefs. One then uses the inference rules to determine what beliefs can be derived from the initial beliefs and the beliefs gained from participating in the protocol. If the set of beliefs is adequate, according to some predefined notion of adequacy, then the protocol is assumed to have been proven correct. If the set of beliefs is not adequate, then it may lead to the discovery of a security flaw in the protocol. This logic, which is meant to be used to prove results about replay attacks in key distribution protocols, was successfully used by its authors to find previously unknown flaws in a protocol that appeared in a draft recommendation for the CCITT X.509 standard [9].

BAN logic is the best known of the modal logics developed for cryptographic protocol analysis. But there are a number of others. These include Bieber's CKT5 [4] and Syverson's KPL [38], both of which reason about knowledge, Rangan's logic of trust [34], which reasons about trust and belief, Moser's logic [29], which reasons about knowledge and belief, and the system of Yahalom, Klein, and Beth [49], which reasons about trust. Syverson's logic can be used to reason about the two kinds of knowledge an intruder may have: knowledge of the word in the sense of seeing a string of bits, versus recognition of the significance of the words. Rangan's logic can be used to reason about the effect of trust in the composition of secure communication channels, and is intended to provide a formal basis for the evolution of belief from trust. The system of Yahalom, Klein, and Beth is used to derive information about the nature of the trust that parties in a protocol must have in each other in order for a protocol to operate correctly. Moser's logic, the only nonmonotonic one considered here, can be used to reason about the way in which beliefs developed through use of cryptographic protocols can be reversed, for example, by learning that a key used in a secure communication was compromised. Bieber's logic, CKT5, can be used to reason about the evolution of knowledge about words used in a cryptographic protocol; like Syverson's logic, it makes a distinction between seeing a message and understanding its significance.

BAN logic has proved to have been by far the most widely used of these logics. Interestingly enough, BAN logic does not attempt to model a protocol in anywhere near the richness as other logics do. BAN does not attempt to model the distinction between seeing a message and understanding it; they are both treated the same way. Likewise, unlike Moser's logic, BAN does not attempt to model the revision of beliefs; the evolution of beliefs in BAN is always monotonic. Moreover, BAN does not attempt to model trust or the lack of it; in BAN logic all principals are assumed to behave according to the rules of the protocol. Finally, since BAN does not attempt to model knowledge, it can not be used to prove results about secrecy; it can be used only to reason about authentication.

BAN's avoidance of these issues is intentional, and it makes for a simple, straightforward logic that is easy to apply and still useful for detecting flaws. This simplicity, combined with its usefulness, is much of the secret of its popularity. However, it also means that the issues it avoids must be addressed in the informal mapping from protocol specification to BAN specification. This has caused some uneasiness among many. For example, Nessett [31] has constructed a counterexample that makes use of the fact that BAN does not reason about secrecy. His example is of a protocol that can be proved to be secure using BAN logic, but is fact divulges a principal's secret key because of bad protocol design. Burroughs, Abadi, and Needham [7] have responded that this example violates one of the assumptions of the logic, namely, that principals do not divulge their secret keys. However, Nessett's example makes the point that this assumption is one that needs to be verified, since keys can be leaked not only by dishonest or incompetent principals, but as the result of the protocol itself.

To show how subtle the reasoning behind the mapping from protocol specification to BAN logic can be, we consider the following protocol due to Aziz and Diffie [2], which was analyzed using BAN logic in their paper. The protocol runs as follows:

1. $\mathrm{A} \hookrightarrow \mathrm{B} : \mathrm{Cert_A},\ \mathrm{N_A},\ \mathrm{other_1}$
2. $\mathrm{B} \hookrightarrow \mathrm{A} : \mathrm{Cert_B},\ \mathrm{K_A(R_B)}, \mathrm{K_B^{-1}}(\mathrm{hash}(\mathrm{K_A(R_B)}, \mathrm{other_2}, \mathrm{N_A}, \mathrm{other_1}))$
3. $\mathrm{A} \hookrightarrow \mathrm{B} : \mathrm{K_B(R_A)}, \mathrm{K_A^{-1}}(\mathrm{hash}(\mathrm{K_B(R_A)}, \mathrm{K_A(R_B)}))$

where $\mathrm{Cert_X}$ is X's public key certificate, $\mathrm{R_A}$ and $\mathrm{R_B}$ are used to construct a session key, $\mathrm{N_A}$ and $\mathrm{R_B}$ are nonces used to guarantee freshness, $\mathrm{other_1}$ and $\mathrm{other_2}$ is information not relevant to this discussion, hash is a hash function, and encrypted messages are formatted in a way that is recognizable by the recipient.

In the idealization of the protocol, the second step is mapped to an assertion that B once said that $\mathrm{R_B}$ was a good key for communication between A and B. But, how does A arrive at this fact? A decrypts the first part of the message and verifies that it is a meaningful message, which A can do since the message has a recognizable format. From the format, and from the fact that it was encrypted with A's key, A concludes that the message was intended for her, and that it is a message saying that $\mathrm{R_B}$ is a good key for communication between A and B. A also verifies the signature on the encrypted message so that she knows that B sent the message. Now she is able to conclude that it was B who said that $\mathrm{R_B}$ is a good key for communication between himself and A.

This reasoning is subtle, and fails if the assumption that the encrypted message is formatted is violated. In that case, one can mount the following attack[1], where I is the intruder, and $\mathrm{I_X}$ denotes the intruder impersonating X:

1. $\mathrm{A} \hookrightarrow \mathrm{B} : \mathrm{Cert_A},\ \mathrm{N_A},\ \mathrm{other_1}$
   (intercepted by I)
2. $\mathrm{I_C} \hookrightarrow \mathrm{B}$: $\mathrm{Cert_C},\ \mathrm{N_A},\ \mathrm{other_1}$
3. $\mathrm{B} \hookrightarrow \mathrm{C}$: $\mathrm{Cert_B},\ \mathrm{K_C(R_B)},\ \mathrm{other_2},\ \mathrm{K_B^{-1}}(\mathrm{hash}(\mathrm{K_C(R_B)}, \mathrm{other_2}, \mathrm{N_A}, \mathrm{other_2}))$
4. $\mathrm{I_B} \hookrightarrow \mathrm{A} : \mathrm{Cert_B},\ \mathrm{K_C(R_B)},\ \mathrm{other_2},\ \mathrm{K_B^{-1}}(\mathrm{hash}(\mathrm{K_C(R_B)}, \mathrm{other_2}, \mathrm{N_A}, \mathrm{other_1}))$

A checks the signature, and applies its private key to $\mathrm{K_C(R_B)}$ to obtain $\mathrm{K_A^{-1}}(\mathrm{K_C(R_B)})$, which she then thinks is the key.

We note that this attack results in at worst a denial of service, since, although the intruder convinces A that a nonkey is a key, the intruder never learns the word that A accepts as a key, and thus cannot impersonate A or B or read encrypted traffic. However, the conclusion of the BAN analysis, that A believes that the word she receives is a good key for communication with B, no longer holds. A no longer has sufficient grounds for drawing that conclusion.

BAN logic will not help its user in distinguishing between the first, correct, version of the protocol and the other, incorrect version. Cases like this and the

[1] This attack was found using the NRL Protocol Analyzer.

Nessett counterexample have caused some concern, and have resulted in various efforts to increase BAN logic's effectiveness. Basically, there are two kinds of approaches that have been taken. One, that followed by Kailar, Gligor, and Gong in [18], is to identify the assumptions that will guarantee that BAN logic is sound if they hold. These assumptions can in turn be verified informally or formally, thus allowing other formal methods and assurance techniques to come to the assistance of BAN logic. The other approach is to increase the scope of BAN logic itself. This is the approach taken by Gong, Needham, and Yahalom in their GNY logic [16], an extended version of BAN logic. This logic includes, among other things, rules for reasoning about message recognizability that makes it possible to reason about a principal's ability to recognize that a bit string is a meaningful message. However, GNY logic is complex, containing over fifty rules, many of them complicated themselves. This has led many to reject this approach as being impractical. It may be, however, that all that is needed is a more systematic approach to the problem. Syverson and van Oorschot [43], for example, have been able, by unifying a number of different logics and developing a common semantics, to simplify them so that they become more tractable, but without sacrificing expressiveness.

4 Using Algebras to Reason About Knowledge

Another approach to applying formal methods to cryptographic protocol analysis is to model the protocol as an algebraic system, similar to the way in which Dolev and Yao model protocols, but to use the algebra to express the state of the participants' (including the intruders') knowledge about the protocol. This is an area that has not received as much attention as the state-machine and logical models discussed above, but the fact that it is able to combine a detailed model as in the state machine approach with an ability to reason about evolution of knowledge comparable to that found in logics of knowledge and belief means, in the opinion of this author, that it merits a closer look.

This approach was first used by Merritt in his PhD thesis [27]. Merritt makes use of *hidden automorphisms* to express an intruder's lack of knowledge about the contents of a message. Suppose, for example, that a principal views a message e(k,m) (denoting the encryption of m with k), where that principal does not know k. Suppose furthermore that we define an automorphism h of the space of words such that h(m) = n for some n, but all other words are left invariant. Then the set of messages known by the principal is invariant under h, (in particular h(e(k,m)) = e(k,m)). Thus effects of the automorphism are invisible to the principal, and can be used to define formally the principal's ignorance of m. Merritt uses this model to prove results about secrecy that are considerably more subtle than the simple secrecy of words; for example, he is able to prove that the correspondence between votes and individual voters in a voting protocol is unknown, even when all the voters and all votes are public.

Another approach to incorporating knowledge into an algebraic model is that taken by Toussaint [44, 45, 46]. In her model the set of words used in a protocol is expressed by an isomorphism between a free algebra with operators encryption and decryption and a crypto-algebra. A participant's state of knowledge is defined by three sets, **F**, **V**, and **SV**. **F** is a set of pairs (a,b) where a is a generator of the free algebra and b is its image in the crypto-algebra. These correspond to words that the principal has seen. **V** (or variables) consists of pairs of the form (x,y) where x is a generator of the free algebra and y is a variable. These correspond to words that the principal is aware of but has not yet seen. **SV** (or semi-variables) consists of pairs of the form (z,a) where a is an element of the crypto-algebra and z belongs to some set of possible encryptions and decryptions. These correspond to such things as the enciphering of messages under unknown keys. The principal knows the structure of the message, but has only limited knowledge about the input. Toussaint shows how this model can be used to describe evolving states of knowledge, and how attacks can be detected by a principal's seeing an inconsistency between messages received and its state of knowledge of the words used in the protocol.

Another approach is to use annotation of words used in algebraic models. This is the approach used by Meadows in [23] to extend the model used by the NRL Protocol Analyzer to reason about protocols that are designed to prevent against attacks in which an intruder may have partial knowledge of the secret words used in a protocol. An example of such an attack would be the case in which a protocol makes use of a password that belongs to a very small key space. A number of protocols have been developed to minimize the bad effects of such passwords, in particular the authentication protocol of Lomas, Gong, Saltzer, and Needham [21]. In Meadows' model each word has a *type* appended to it, which represents the knowledge the intruder has about a word. Some types are subtypes of other types, and thus reflect the intruder's increasing knowledge about the word. Thus a word may be of type possible password, meaning that it belongs to the space of possible passwords, or it may be of type password, meaning that the intruder knows that it is a password. Reduction rules are defined for types as well as words, and general rules of inference for deriving types are given. It is then shown how this approach can be used to model the guessing of a password, and how the Lomas-Gong-Saltzer-Needham protocol can be modeled.

Research in this area has not been as active as research in developing and applying state-machine models and logics of knowledge and belief. However, the models' success in representing very subtle kinds of knowledge, and the fact that the objects modeled correspond strongly to entities and messages used in the tools based on state machines suggest that these models could be used to provide the state machine tools with a stronger capability of modeling the knowledge that an intruder could gain. As yet, little research has been done on this problem. In [23] Meadows attempted to incorporate her extension of NRL Protocol Analyzer model into the tool itself, but the result was considered unsatisfactory because of the difficulty of modeling rules for increasing an intruder's knowledge as the kinds of reduction rules acceptable by the NRL Protocol Analyzer. However, the general question of whether and how these algebras can be incorporated in state machine analysis tools is still an open one.

5 Model Granularity and Range

It is unlikely that any formal method will be able to model all aspects of a cryptographic protocol, and thus it is unlikely that any formal method will be able to detect or prevent all types of protocol flaws. The best we can hope for is that it will be able to guarantee that the protocol is correct given that a certain well-defined set of assumptions is satisfied. Thus, for example, most formal models make the assumption that the underlying cryptosystem is perfect, that is, that an intruder can gain no information about a message that was encrypted with a key that he does not know. However, it is not always clear what we should attempt to include in the model, and what should be included in the assumptions. As we have seen in the discussion in the section on applying logics of knowledge and belief, much of the controversy over BAN logic concerns what should be addressed by the logic and what should be left as assumptions to be verified by other means.

In general, we can state three criteria that should be satisfied when deciding whether a feature should be included in a model:

1. Is it possible to include the feature and still have the analysis be tractable?
   For instance, although nonmonoticity can be considered a feature of cryptographic protocols, most logics of knowledge and belief that are applied to these protocols are monotonic, since monotonic logics are in general more tractable than nonmonotonic ones.
2. How useful is the ability to model the feature?
   Does the feature affect security? Is the feature likely to fail? Can the feature be handled in ways that make formal analysis unnecessary, or is the feature handled by other formal methods?
3. How well defined and natural is the boundary between the features modeled and the features not modeled?
   For example, a model that included an intruder's ability or inability to take discrete logarithms but did not include the ability or inability to factor would be considered somewhat lopsided, since these two problems are closely related.

Different systems may choose the cut-off point at different places. For example, logics such as BAN can be thought of as reasoning about the *intent* of messages. The verification that a message performs its intended function is done when the user of BAN logic maps lines in a protocol specification to their idealizations. Burrows, Abadi, and Needham describe the assumptions that will help to guarantee that this idealization will be correct, but the verification that the assumptions hold, or that they guarantee the correctness of the mapping, is not part of the logic.

State-machine-based systems based on the Dolev-Yao model, such as the Interrogator or the NRL Protocol Analyzer, generally give a more detailed approach. Messages are represented as concatenations of abstract message fields, and properties of cryptographic systems that are necessary to the correct operation of a cryptographic protocol, such as the fact that encryption and decryption with the same key in a shared-key system cancel each other out, are also modeled. However, properties of cryptosystems that may affect the security of a protocol, such as the commutative-associative property of exclusive-or, or the homomorphic properties of RSA, are usually not modeled (with a few exceptions: see for example [14]). Cryptographic integrity mechanisms are also usually not explicitly modeled. It is assumed that secrecy and integrity mechanisms do their job, but it is not asked exactly how the job is done.

It is possible to construct useful models at a lower level of granularity than this. For example, In [36] Stubblebine and Gligor introduce a model that allows them to model such things as cryptographically weak checksums (in which, given a checksum of a message, it is possible to produce another message that evaluates to the same checksum) versus collision-free checksums (in which, given a message and its checksum, it is computationally infeasible to produce another message with the same checksum). Given this model, they were able to uncover a flaw in the Kerberos system that was the result of its using such a weak checksum. The flaw was subtle and involved an intruder's cutting and pasting together different messages, and disguising the fact that he had done so by his ability to produce messages that evaluated to the same checksum. Their approach was also used to find a flaw in Privacy-Enhanced Mail [37]. Stubblebine's and Gligor's success in detecting these flaws shows that we still have not reached the limits of the degree of detail which which we can model a cryptographic protocol and still have fruitful results.

One might be tempted to conclude that only the most detailed models are useful. But all these models at the various levels of abstraction have their areas of usefulness. In general, it is most helpful to use the more abstract models at earlier points in the design stage, when implementation details have not been yet decided upon. For example, a protocol designer might use BAN or one of the similar logics to determine what the role of each message of a protocol should be. He or she might then use a state-based tool when attempting to determine what the structure of messages should be. Finally, when the actual implementation, including formatting of messages, choice of encryption systems, and choice of integrity mechanisms, is in question, it would be most appropriate to use something like the Stubblebine-Gligor model to determine how these implementation decisions affect the security of the protocol. Such an approach would allow us to locate errors as early as possible with a minimum amount of work.

6 Requirements Modeling

An area that is beginning to be explored in more depth is the question of how we specify the correctness of a protocol in the first place. Early work on applying formal methods to protocol analysis concentrated on secrecy, by attempting to show that an intruder could not learn a particular word or words. This was quickly realized to be inadequate, since many cryptographic protocols provide services such as authentication, that are only indirectly related to secrecy. At this point, it becomes necessary to determine exactly what the goals of a secure protocol must be.

This problem has been approached from several different angles, some with the aim of developing a set of criteria that can be applied to protocols in general, and others with the aim of developing ways to express criteria for a number of different types of protocols. In [11] Diffie, van Oorschot, and Wiener developed informal requirements for the correctness of an authentication protocol. Briefly, they say that session keys should remain secret, and that protocol runs should match. The latter means that, if A and B participate in a run of a protocol, then A's record of messages received from B matches B's record of messages sent to A, and vice versa. This notion has been formalized by Bellare and Rogaway in [3], using a model based on communicating probabilistic Turing machines. In the Bellare-Rogaway model, certain failure events, such as the compromise of old session keys, are included, so that the protocol can still be shown to satisfy matching runs in the face of these failures. Diffie, van Oorshot, and Wiener's notion of matching runs has also been formalized by Syverson in his extension of the Abadi-Tuttle logic to include temporal formalisms [39].

In [47] Woo and Lam independently take a similar approach to defining security of key distribution protocols. In this work Woo and Lam define a semantic model characterizing protocol executions. Two basic security properties, correspondence and secrecy, are defined. Secrecy is self-explanatory, while correspondence pertains to the requirement that certain events can take place only if others have taken place previously. The notion of correspondence thus bears a resemblance to the notion of matching protocol runs, but it is broader, since the two events in question do not have to be the sending and receiving of the same message. Woo and Lam show how to specify authentication protocol requirements in terms of assertions about correspondence and secrecy. Like Bellare and Rogaway, they also specify failure events as part of their model.

Another approach to specifying protocol requirements is shown in the requirements language currently being developed for the NRL Protocol Analyzer [40]. The requirements specified in this language have a form similar to the notion of correspondence of Woo and Lam, in that the requirements are given on sequences of events. The difference is that, instead of giving a general requirement for correspondence that applies to all protocols, the user of the language specifies only the requirements that are necessary for protocols belonging to a particular class to perform their intended functions. Thus requirements will vary according to the intended function of a protocol. The notion of "event" is also somewhat broader than that of Bellare and Rogaway or Woo and Lam in that any state change can be an event. Thus an intruder's learning a word is modeled as an event. This means that it is unnecessary to define secrecy as a separate part of the model. Internal state transitions can also be modeled as events. This makes it possible to model such things as a timestamp becoming obsolete according to a principal's internal clock. As in the Woo-Lam model, failure such as key compromise can also be modeled as events. In [40] Syverson and Meadows give a set of requirements for various kinds of message authentication protocols, while in [41] they give a set of requirements for key distribution protocols with reauthentication.

In [48] Yahalom takes a similar approach to Syverson and Meadows, but with different goals. Like them, he describes the various message passing events that must take place in a key distribution protocol, and states requirements on a protocol in terms of which events must occur before others. However, he uses these requirements to determine the minimum number of messages that must be sent in order for a protocol to satisfy these requirements, and then constructs a protocol that uses this number of messages. Thus his goals are to use the formulation of protocol requirements to achieve greater performance within the bounds set forth by the requirements.

Most work on formalizing security requirements for cryptographic protocols has concentrated on key distribution protocols. However, work on applying this approach to other areas has begun to appear. In [32] Pfitzmann uses a formal specification language to specify requirements for different kinds of signature schemes. The goal of her work is provide a classification system for the various kinds of signature schemes and their security requirements.

We note that there is still much work that remains to be done on security requirements for cryptographic protocols, and that this work could have far-reaching implications. Protocols have been developed for such applications as software protection, secure sharing of resources, and secure transmission of authorization. In general, if any two components of a distributed system engage in a transaction using a hostile communication medium, then they must make use of a cryptographic protocol to enforce their security requirements. Thus the topic of security requirements for cryptographic protocols is very close to the topic of security models for distributed systems. Once we start talking about requirements for cryptographic protocols we begin to start talking about what part of the system supplies what kind of security service to the other parts, what parts trust other parts and in what way, and so on. These are the very kinds of issues that must be addressed when we consider the security requirements for distributed systems in general. Thus in future years we can probably expect the area of cryptographic protocol analysis and the area of security modeling for distributed systems to grow closer and closer together.

7 The Use of Formal Methods in the Design of Protocols

Most of the existing work in the application of formal methods to cryptographic protocols has been concentrated on applying the methods to the analysis of existing protocols. However, in the long run it would be cheaper and more effective to use the methods in the design of the protocol in the first place, and so save the expense of redesign. In general, not much research has been done in this particular area. But, we believe that this is mainly because of the youth of the field. The use of formal methods in design as well as analysis is a natural application of the technology, and we can expect to see more of it in cryptographic protocols. In this section we describe some work that has been done so far.

The incorporation of formal methods into design can be done in two ways, as is the case with the incorporation of formal methods into the design of any product. One approach is to develop methodologies for design of protocols so that they will be more amenable to analysis by formal methods. This is the approach taken by Heintze and Tygar in [17]. In that paper they develop a modular approach to designing cryptographic protocols, and set forth properties of modules that will guarantee that their composition satisfies the desired security properties.

The other approach, which can be used together with the first, is a layered approach, in which a relatively abstract model is used at the top layer, and each succeeding layer is proved to be an implementation of the layer above it, until finally either a detailed specification or the actual protocol code is produced. This would be a more formal version of the strategy of using increasingly detailed models that was discussed in the section on model granularity. Much of the work on requirements specification, such as the Syverson and Meadows work that we discussed in the last section, has this flavor. Also, for the application of BAN logic, Carlsen [8] has developed a parser that will translate members of a limited class of protocol specifications into BAN logic. The option of integrating existing tools and methods that use models of different granularity is also an attractive one. Care must be taken, however, that the models underlying the methods can be made compatible. For example, in [26] Meadows develops a model of computation for the NRL Protocol Analyzer and compares it with the one Abadi and Tuttle [1] have developed for BAN logic. This is used to point out several important differences in the assumptions made by the two models that would have to be addressed before they were integrated.

8 Conclusion

In this paper we have attempted to give an overview of the state of the art in the application of formal methods to cryptographic protocol analysis. It is still a young field, so it is not possible to draw any final conclusions about the way in which it is headed. However, we have been able to identify some major trends and subfields, as well as identify some areas in which we believe further research is needed. We hope that in future years it will continue to build on its successes, and as it matures, will become a useful part of the secure systems designer's toolbox.

References

1. Martín Abadi and Mark Tuttle. A Semantics for a Logic of Authentication. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, August 1991.

2. A. Aziz and W. Diffie. Privacy and Authentication for Wireless Local Area Networks. *IEEE Personal Communications*, 1(1):25–31, 1994.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology - CRYPTO '93*, volume to appear. Springer-Verlag, 1994.
4. P. Bieber. A Logic of Communication in a Hostile Environment. In *Proceedings of the Computer Security Foundations Workshop III*, pages 14–22. IEEE Computer Society Press, June 1990.
5. J. Burns and C. J. Mitchell. A Security Scheme for Resource Sharing Over a Network. *Computers and Security*, 19:67–76, February 1990.
6. Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, February 1990.
7. Michael Burrows, Martín Abadi, and Roger Needham. Rejoinder to Nessett. *Operating Systems Review*, 24(2):39–40, April 1990.
8. U. Carlsen. Generating Formal Cryptographic Protocol Specifications. In *Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 137–146. IEEE Computer Society Press, May 16-18 1994.
9. CCITT. *CCITT Draft Recomendation X.509. The Directory-Authentication Framework, Version 7*, November 1987.
10. D. E. Denning and G. M. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):198–208, 1981.
11. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes, and Cryptography*, 2:107–125, 1992.
12. D. Dolev, S. Even, and R. Karp. On the Security of Ping-Pong Protocols. *Information and Control*, pages 57–68, 1982.
13. D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.
14. Shimon Even, Oded Goldreich, and Adi Shamir. On the Security of Ping-Pong Protocols When Implemented Using the RSA. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, pages 58–72. Springer-Verlag, 1985.
15. Joan Feigenbaum. Overview of Interactive Proof Systems and Zero Knowledge. In G. J. Simmons, editor, *Contemperary Cryptology: The Science of Information Integrity*, chapter 8, pages 423–439. IEEE Press, 1991.
16. L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *IEEE Computer Society Symposiun in Security and Privacy*, pages 234–248. IEEE Computer Society Press, May 1990.
17. Nevin Heintze and J. D. Tygar. A Model for Secure Protocols and Their Compositions. In *1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–13. IEEE Computer Society Press, May 1994.
18. R. Kailar, V. D. Gligor, and L. Gong. On the Security Effectiveness of Cryptographic Protocols. In *Proceedings of the Fourth Internation Working Conference on Dependable Computing For Critical Applications*, 1994. to appear.
19. Richard Kemmerer. Using Formal Methods to Analyze Encryption Protocols. *IEEE Journal on Selected Areas in Communication*, 7(4):448–457, 1989.
20. Richard Kemmerer, Catherine Meadows, and Jonathan Millen. Three Systems for Cryptographic Protocol Analysis. *Journal of Cryptology*, 7(2), 1994.
21. T. M. A. Lomas, L. Gong, J. H. Saltzer, and R. H. Needham. Reducing Risks From Poorly Chosen Keys. *Operating Systems Reviews*, 23(5):14–18, 1989.
22. D. Longley and S. Rigby. An Automatic Search for Security Flaws in Key Management Schemes. *Computers and Security*, 11(1):75–90, 1992.

23. C. Meadows. Representing Partial Knowledge in an Algebraic Security Model. In *Proceedings of the Computer Security Foundations Workshop III*, pages 23–31. IEEE Computer Society Press, June 1990.

24. C. Meadows. A System for the Specification and Analysis of Key Management Protocols. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 182–195. IEEE Computer Society Press, Los Alamitos, California, 1991.

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

26. C. Meadows. A Model of Computation for the NRL Protocol Analyzer. In *Proceedings of the 7th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 1994.

27. M. J. Merritt. Cryptographic Protocols. Ph.d. thesis, Georgia Institute of Technology, 1983.

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

29. L. Moser. A Logic of Knowledge and Belief for Reasoning About Computer Security. In *Proceedings of the Computer Security Foundations Workshop II*, pages 57–63. IEEE Computer Society Press, June 1989.

30. 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.

31. D. M. Nessett. A Critique of the Burrows, Abadi, and Needham Logic. *Operating Systems Review*, 24(2):35–38, April 1990.

32. Birgit Pfitzmann. Sorting Out Signature Schemes. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 74–85. ACM SIGSAC, ACM, November 3-5 1993.

33. G. B. Purdy, G. J. Simmons, and J. A. Studier. A Software Protection Scheme. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 99–103. IEEE Computer Society Press, April 1982.

34. P. V. Rangan. An Axiomatic Basis of Trust in Distributed Systems. In *Proceedings of the 1988 Symposium on Security and Privacy*, pages 204–211. IEEE Computer Society Press, April 1988.

35. G. J. Simmons. How to (Selectively) Broadcast a Secret. In *Proceedings of the 1985 Symposium on Security and Privac*, pages 108–113. IEEE Computer Society Press, April 1985.

36. 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.

37. S. Stubblebine and V. Gligor. Protecting the Integrity of Privacy-Enhanced Mail. In *PSRG Workshop on Network and Distributed System Security*, pages 75–80, February 11-12 1993.

38. P. Syverson. Formal Semantics for Logics of Cryptographic Protocols. In *Proceedings of the Computer Security Foundations Workshop III*, pages 32–41. IEEE Computer Society Press, June 1990.

39. Paul Syverson. Adding Time to a Logic of Authentication. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 97–101. ACM SIGSAC, ACM, November 3-5 1993.

40. 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.

41. Paul Syverson and Catherine Meadows. Formal Requirements for Key Distribution Protocols. In *Proceedings of Eurocrypt '94*, 1994. to appear.
42. Paul F. Syverson. Knowledge, Belief, and Semantics in the Analysis of Cryptographic Protocols. *Journal of Computer Security*, 1(3):317–334, 1992.
43. Paul F. Syverson and Paul 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.
44. M.-J. Toussaint. Verification of Cryptographic Protocols. Ph.d. thesis, Universite de Liege (Belgium), 1991.
45. M.-J. Toussaint. Deriving the Complete Knowledge of Participants in Cryptographic Protocols. In *CRYPTO '91 (Advances in Cryptology—CRYPTO '91)*. Springer-Verlag, 1992.
46. M.-J. Toussaint. Separating the Specification and Implementation Phases in Cryptology. In *Computer Security - ESORICS 92*, volume LMCS 638, pages 77–102. Springer-Verlag, 1992.
47. T. Y. C. Woo and S. Lam. A Semantic Model for Authentication Protocols. In *Proceedings of the 1993 Symposium on Research in Security and Privacy*, pages 178–194. IEEE Computer Society Press, May 1993.
48. R. Yahalom. Optimality of Asynchronous Two-Party Secure Data-Exchange Protocols. *Journal of Computer Security*, 2(2-3):191–209, 1994.
49. R. Yahalom, Birgit Klein, and Thomas Beth. Trust relationships in secure systems: A distributed authentication perspective. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, pages 150–164. IEEE Computer Society Press, May 1993.

This article was processed using the LaTeX macro package with LLNCS style