Ordering from Satan’s menu: a survey of requirements specification for formal analysis of cryptographic protocols

Catherine Meadows

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

Received 13 June 2003; received in revised form 21 November 2003; accepted 11 December 2003

Abstract

The application of formal methods to cryptographic protocol analysis has been a growth area recently. Most of the attention has been paid to the design of languages for the specification of cryptographic protocols and algorithms for evaluating their security. However, the ability to specify their desired behavior correctly is also important; indeed many perceived protocol flaws arise out of a misunderstanding of the protocol’s requirements. In this paper, we give a survey of research in requirements specification for formal analysis of cryptographic protocols. We start with a brief history of the use of requirements specification for cryptographic protocols. We then outline some of the main current trends and areas of research. We conclude with a discussion of some open problems.

© 2003 Published by Elsevier B.V.

Keywords: Cryptographic protocols; Formal methods; Requirements engineering

1. Introduction

Cryptographic protocols must be able to process transactions securely in face of an intruder who may have complete control of a network, that is, who may be able to monitor, delete, alter, or redirect traffic, who has access operations such as encryption available to legitimate principals, and who may be in league with one or more legitimate but dishonest principals. As a result, it is widely acknowledged that it is difficult to design such protocols correctly. Indeed, Anderson and Needham [50] have compared

E-mail address: meadows@itd.nrl.navy.mil (C. Meadows).
the problem to that of “Programming Satan’s Computer,” that is, writing a program on a computer that is capable of returning subtly incorrect answers designed to subvert the program’s goals.

On the other hand, it has often been pointed out that, although it is difficult to get cryptographic protocols right, what is really difficult is not the design of the protocol itself, but of the requirements. Many problems with security protocols arise, not because the protocol as designed did not satisfy its requirements, but because the requirements were not well understood in the first place. Not surprisingly, the realization of this fact has lead to a considerable amount of research in the formulation and formalization of security requirements for cryptographic protocols. However, most of this literature is scattered, and unlike the topic of cryptographic protocol analysis in general, there is little existing survey work providing roadmaps to readers interested in learning more about the topic. In this paper, we attempt to remedy this deficiency by providing a brief history and survey of the work that has been done in this area, and outlining what we consider to be some of the open problems.

Any scheme for expressing requirements should satisfy three properties:

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

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

The paper is organized as follows. We begin Section 2 by describing some of the early approaches to the formal specification of cryptographic protocol requirements, including that of Burrows, Abadi, and Needham. In Section 3, we describe some of the main current approaches to requirements in terms of a spectrum from extensional to intensional requirements. We also discuss two topics of particular interest: requirements specification in the popular strand space theory, and the use of non-interference to provide a framework for correspondence properties. In Section 4, we discuss several emerging areas of research: graphical languages for specifying cryptographic protocol requirements, requirements for anonymity systems, requirements for protocol properties other than safety, and expression of quantitative requirements. In Section 5, we sum up what we believe to be some of the open problems, and conclude the paper.

We offer one caveat here: we are not offering a survey of formal analysis of cryptographic protocol in general, but only as it pertains to the formulation and formalization of requirements. Thus, we leave out much important work in the area in general (for example, the recent valuable work in complexity and decidability of crypto protocol analysis, and much of the work on developing specialized languages, formal systems,

---

1 We note, however, that the recently published book by Boyd and Mathuria [6] provides an exception to this, covering much of the same ground that we do in Section 3.1 of this paper.
and tools), and concentrate mostly on work in which different types of requirements were first identified and formalized.

2. Early work in cryptographic protocol requirements

Most of the existing approaches to applying formal methods to cryptographic protocol analysis stem ultimately from that of Dolev and Yao [14], who developed the first formalization of the intruder model that is commonly used today. However, since Dolev and Yao’s work and its immediate successors were mainly focused on theoretical results about the complexity of cryptographic protocol analysis, only one type of requirement was considered, and that was the simplest: that some term or set of terms designated as secret should not be learned by the intruder. Some of the earlier work on automated cryptographic protocol analysis, such as the first versions of the Interrogator [35], also restricted itself to this limited definition of secrecy. Others, such as the earlier versions of the NRL Protocol Analyzer (NPA) [30], allowed the user to specify security in terms of the unreachability of insecure states. The user could specify a state by specifying terms known by the intruder and values of local state variables. Thus, one might specify a state in which the intruder knew a secret key by describing the state as one in which the intruder knew the value of a state variable local to an honest principal that held a key used for communication with another honest principal. However, the user was not given any further assistance in constructing requirements.

Probably, the first formal cryptographic protocol analysis system to provide a real mechanism for constructing formal requirements was the belief logic of Burrows, Abadi, and Needham (BAN) [7].

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

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

One uses BAN logic by attempting to see which of these beliefs can be derived from an idealization of the protocol. The BAN logic does not dictate which beliefs a protocol should be able to satisfy; rather it is up to the protocol analyst to decide what beliefs a protocol should guarantee, and to determine if those beliefs can be derived from the protocol. Thus, one might require that Alice believes that K is a good key for communicating for Bob, and that Bob believes that K is a good key for communicating with Alice, but one might or might not want to require that Alice believes that Bob believes that K is a good key for communicating with Alice, and vice versa. Thus, BAN logic provides what it probably the first formal system for specifying cryptographic protocol requirements.
3. Formulating and expressing standard cryptographic protocol requirements

3.1. Secrecy and correspondence

In the early to mid-1990s, the approach to cryptographic protocol verification tended towards the application of general-purpose tools such as model checkers and theorem provers. With this came the need to develop means for specifying the properties one was attempting to prove. Since, in general, researchers were now reasoning directly about messages passed in a protocol, rather than about beliefs that were developed as a result of receiving those messages, it now made sense to develop requirements in terms of messages sent and received rather than beliefs derived.

As is the case for requirements in general, requirements for cryptographic protocols tend to fall into two categories, extensional and intensional. Extensional systems provide a small set of generic requirements that can be defined independently of the details of any particular protocol. Intensional systems provide languages and techniques that can be used to specify requirements for specific protocols in terms of the protocols themselves. This concept was first discussed in detail in the context of cryptographic protocols by Roscoe in [39]. He noted that the earlier work in cryptographic protocol requirements, such as BAN, leaned to the extensional side, and he showed how one might specify intensional protocol requirements in CSP.

Requirements for cryptographic protocols also fall into two classes that are related to the properties that such protocols are intended to enforce: secrecy and correspondence. Secrecy requirements describe who should have access to data. Correspondence requirements describe dependencies between events that occur in a protocol, and are usually used to express authentication properties. These two types of requirements later turned out to be more closely related than one might think (both Syverson and Meadows [46] and Schneider [40] define secrecy requirements as a type of correspondence requirement), but for the moment we shall treat them as separate.

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

At first, correspondence requirements appeared to be the most subtle and complex. Thus, the earlier work tended to concentrate on these. Moreover, the emphasis was on extensional requirements and the ability to characterize a general notion of correspondence in a single definition. Probably the first work in this area was that of Bird et al. [5]. In the introduction to their paper, they describe an error-free protocol run between two principals A and B to be one in which all executions viewed by both parties match exactly one-to-one. This idea is refined by Diffie et al. [13] to the idea of matching protocol runs, which says that at the time Alice completes a protocol the other party’s record of the run matches Alice’s. This notion was further refined and formalized by
Bellare and Rogaway [4] to the notion of matching conversations, which developed the idea in terms of a complexity-theoretic framework.

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

To see what we mean, we consider the attack found by Lowe [27] on the Station-to-Station protocol of [13]. The protocol is defined as follows. We use the standard notation for representing cryptographic protocols. $A \rightarrow B : M$ means that $A$ sends message $M$ to $B$, $E_K(X)$ stands for $X$ encrypted with key $K$. $S_B(X)$ stands for $X$ signed with $B$’s private key.

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

Lowe’s attack runs as follows:

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

Thus, at the end of $A$’s run, $A$ believes that it shares a key with $B$. $B$, however, thinks that $C$ is trying to establish a connection with it, and it will reject $A$’s final message when it receives it, because it is expecting confirmation from $C$, not $A$. On the other hand, the protocol does satisfy the matching protocol runs definition of security, since $A$’s picture of the authenticated portions of the messages is the same as $B$’s. Indeed, this is the protocol used to illustrate the concept by Diffie et al. [13].

Lowe’s attack, of course, does not mean the Station-to-Station protocol is insecure. (Indeed, this very feature of that protocol is seen as a desirable property in the latest version of IKEv2, the proposed replacement to the Internet Key Exchange protocol [22]; we will discuss the implications of this in more detail later on). All it does is to show that, if the name of the intended recipient is not included in the responder’s message, a definition of security that is specified in terms of conditions on correspondence between messages will not catch lack of agreement on information that is never sent.

Lowe’s solution to this problem in [27] was to strengthen the matching protocol runs requirement to include the condition that when $A$ completes a protocol run with $B$, then not only should the two protocol runs match, but $B$ should believe that he has been running the protocol with $A$. In a later paper [28], he developed this idea further, developing a hierarchy of authentication requirements which gave conditions of varying degrees of strictness on the conclusions a principal $A$ could draw about $B$’s view of the protocol after completing the protocol with $B$. These were then formalized using the process algebra CSP.

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

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

This is the sort of reasoning that lay behind Syverson and Meadows’ development of a requirements language for the NRL protocol analyzer [46], which eventually became known as the NRL protocol analyzer temporal requirements language (NPATRL). The idea is to develop a simple temporal language that can be used to specify the type of requirements that are commonly used in authentication and key distribution protocols. The atomic components of the language correspond to events in the protocol (e.g. the sending and receiving of messages, or the intruder’s learning a term). Besides the usual logical connectives, it contains only one temporal operator, $\circ$, or “happened previously.” The use of this single logical operator reflects the fact that most correspondence requirements can be expressed in terms of events that must have or must have not occurred before some other events.

Although NPATRL is a very simple language, we have found it to be useful for specifying some widely varying types of cryptographic protocols. These include key distribution and key agreement protocols [47,48], complex electronic commerce protocols such as SET [32], and, most recently, group key distribution protocols [33].

One interesting result is that NPATRL has turned out to be useful for specifying complex secrecy requirements as well as complex authentication requirements. Early requirements for secrecy simply designated some information, such as keys, as secret, and all that needed to be guaranteed was that these keys would not be available to an intruder. However, more recently, requirements such as perfect forward secrecy put other conditions on an intruder learning a term. Perfect forward secrecy requires that, if a master key is compromised, then an intruder can only learn a session key if it was generated after the compromise, not before. This property is satisfied, for example, by protocols that make use of authenticated Diffie–Hellman, such as the Station-to-Station protocol. If the keys used to generate the digital signatures in that protocol are compromised, then an intruder can mount a man in the middle attack on the protocol to trick principals into accepting new keys which it knows. But it cannot use its knowledge of the signature keys to help in the compromise of already generated and accepted keys. Such a requirement, in which the temporal relationship between two types of key compromise is paramount, is straightforward to specify using a temporal language.

However, temporal systems are not necessary to specify most standard correspondence and secrecy properties. By now, it is safe to say that it is standard for any system developed for cryptographic protocol analysis to have the ability to express both secrecy and correspondence properties, where secrecy means that the intruder does not
learn certain data, and correspondence means that if a certain event occurs, then certain other events occurred previously. A study of the way different formal systems handle these requirements would be illuminating (for example, different systems allow different amount of expressiveness in describing the order in which events preceding the final event may occur, as well as different levels of support in expressing complex requirements); however, space does not allow us to go into more detail about that here. In the next two sections, we restrict ourselves to two special cases: strand spaces, because they have become an extremely popular language for specifying protocols and their requirements, and non-interference, because it offers a framework in which a whole class of correspondence requirements can be represented.

3.2. Strand spaces

A discussion of intensional specifications in cryptographic protocol analysis would not be complete without a discussion of strand spaces. This is a formalism that was mainly intended to facilitate protocol specification and analysis, but it has useful applications for requirements specification as well. Strand spaces [17] are well-known and popular models for cryptographic protocol analysis, in which the actions of principals are modeled in terms of graphs. A strand represents a principal executing a role in a protocol. The sending and receiving of messages is represented by positive and negative nodes. Nodes that represent one event immediately preceding another on a strand are connected by double arrows. A bundle is a collection of strands, in which positive send nodes can be connected to negative receive nodes via a single arrow if the message sent matches the message received. For example, we include a strand space representation of the initiator’s role in the Needham–Schroeder public key protocol, using $K_X$ to stand for $X$’s public key (Fig. 1).

If this had been represented in the conventional notation we used earlier for the Station-to-Station protocol, it would appear as follows:

1. $A \rightarrow B : K_B(N_A, K_A)$,
2. $B \rightarrow A : K_A(N_A, N_B)$,
3. $A \rightarrow B : K_B(N_B)$.

One advantage of strand spaces is that one can characterize a principal’s execution of a role in an instance of a protocol in terms of a single strand, parametrized with
appropriate data values. For example, the Needham–Schroeder protocol requires an initiator $A$ to send a nonce $N_A$ to a principal $B$, and to receive a nonce $N_B$ from $B$, and $B$ to do likewise with respect to $A$. We could represent this by two parametrized stands: $\text{INIT}[A, B, N_A, N_B]$ and $\text{RESP}[B, A, N_B, N_A]$, and say that the protocol is correct if for every bundle containing a strand $\text{INIT}[A, B, N_A, N_B]$ there is one and only one instance of $\text{RESP}[B, A, N_B, N_A]$.

The originators of strand spaces did not provide a formal requirements specification language, but we note that others who have made use of strand spaces have, see for example Song’s Athena model checker [45]. Parametrized strands may provide the most succinct way of specifying a large class of intensional requirements for cryptographic protocols. They do, however, lose some expressiveness, since it is difficult to specify which individual actions (e.g. sends, receives) should precede which others. In the section on graphical requirements, we will discuss how this capability might be reincorporated.

### 3.3. Non-interference

Non-interference was originally developed for the study of security in multilevel secure systems, that is, systems that must protect data classified at different security levels. One goal of such a system is to prevent data classified at high to processes who have legal access to data classified at low via covert channels. We say that a system is non-interfering if the behavior visible to a low process is the same whether or not any high processes are present. Non-interference, as originally defined by Goguen and Meseguer [19], was formulated in terms of conditions on traces, but it has also been couched in terms of state machines, process algebras, and logics of knowledge and belief (for the last, see Halpern and O’Neill [21]).

Non-interference might seem to be an odd choice for modeling encryption protocols, since it does not really appear to be that compatible with encryption. But non-interference is used in a somewhat unexpected way by Durante et al. [15]. Instead of identifying the secret data as high, one identifies the intruder with the high process and the honest principals with the low processes. Loosely speaking, a protocol is secure if the low behavior, defined in terms of initial initiator events and final initiator and responder events, is the same whether or not the intruder is present.

The non-interference approach to security requirements is not only highly extensional, but it may also be overly restrictive. For example, it is more restrictive than matching conversations. It is easy to see that if a protocol satisfies non-interference it will satisfy matching conversations, since matching conversations is achieved in the absence of an intruder. On the other hand, the Station-to-Station protocol achieves matching conversations, but not non-interference.

More recently, Focardi et al. [18] have developed a more inclusive system for specifying requirements called generalized non-deducibility on composition, or $\text{GNDC}$. It is defined as follows.

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

$$(P\parallel X)\setminus C \approx x(P).$$

In the case that $x$ is the identity function and $\approx$ is trace equivalence, the property becomes $\text{NDC}$, or non-deducibility on composition, which requires that the traces produced by the process in composition with an intruder be the same as the traces produced by the process in the absence of the intruder. This can be thought of as an information-flow property in which the intruder and $P$ play the part of high and low, respectively, corresponding to the standard multilevel application of non-interference for multilevel security [19]. $\text{NDC}$, since it requires that a process behave in the presence of an intruder exactly as it would behave in the absence, is more stringent than any of the other requirements that have been discussed in this section.

Moreover, $\text{GND}_\approx$ provides a framework that allows one to specify less restrictive requirements such as the various forms of correspondence discussed earlier, and the types of requirements that would be defined in a temporal language such as NPATRL. Thus, $\text{GND}_\approx$ can be thought of as providing a general framework for requirements, including requirements that go beyond the usual notions of correspondence, such as liveness. Note also, that $\text{GND}_\approx$ and, in particular, $\text{NDC}$ are not safety properties, although in the case that the number of traces is finite, it is possible to check whether $\text{NDC}$ holds by comparing the sets of traces produced by both processes. This means that it should also be possible to use $\text{GND}_\approx$ to specify liveness properties for cryptographic protocols, about which we will learn more in Section 5.

We note also that non-deducibility on composition is not necessarily the strongest possible definition of security. Our candidate for that would be Gong and Syverson’s [20] fail-stop property. It basically says that at any point in a protocol’s execution, if it deviates from its normal behavior, then that deviation is detected and the execution is halted. This is a stronger requirement than any of the others that we have seen, since it puts conditions on partial executions as well as completed ones. We conjecture that it should be straightforward to incorporate the fail-stop definition of correctness into the $\text{GND}_\approx$ framework by incorporating all send and receive events in the model instead of just initial and final events.

4. Graphical requirements languages

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

The first of these is known as strand space pictures [16], based on the strand space model described earlier in Section 3. This model facilitates the graphical representation
of protocols, and [16] actually describes a number of ways in which the graphical
features of strand spaces could be used. But the one of most interest to us is the way
in which they can be used to represent requirements. Using strand space representation
of protocols, it is possible to represent correspondence requirements in terms of relative
placement of strands. Thus, if we want to specify a correspondence requirement which
requires that if certain messages are accepted, then other messages were sent previously,
we can represent sending and receipt of the messages we are interested in by portions
of strands, and we can use the placement of the strands (so that earlier nodes appear
above later ones) to indicate which events we want to occur before others. This not
only gives us a convenient graphical way of expressing requirements, but is somewhat
more expressive than the more common use of expressing requirements in terms of
conditions on the existence of parametrized strands.

The strand space pictures methodology, was never, as far as we know, developed into
a full-fledged procedure with well-defined ways for representing major classes of re-
quirements. However, in [16] the authors give several examples which show how some
standard requirements such as freshness or agreement properties could be represented
in this framework.

A somewhat different approach has been taken by Cervesato and Meadows [10] in the
development of a graphical representation of the NPATRL language. This representation
was based on the fact that queries in the NRL Protocol Analyzer, for which NPATRL
was designed, are couched in terms of events that should or should not precede some
specified event. Such a way of formatting queries has an obvious connection to fault
trees. A fault tree is a graphical means of representing failure modes in safety-critical
systems. The root of the tree represents the failure with which the system designer is
concerned, and the branches represent the conditions under which the fault can occur.
The main difference between NPA queries and fault trees is that in NPA queries the
relationship is one of precedence, while in fault trees it is one of causality. Otherwise
the structure is very similar. Moreover, the graphical representation makes it easier to
understand the relationships between the various events. For this reason, it was found
helpful, in particular, to represent the GDOI requirements, especially the more complex
ones, in terms of fault trees. In [10], a fault tree semantics for the subset of NPATRL
requirements accepted by the NPA is developed, and some sample requirements are
shown.

To see how the approach would work, consider the two specifications given in
Fig. 2. They describe two types of freshness for the GDOI group key. The arrows
map to “implies happened before,” the straight lines with a cross map to “happened
before,” while the unadorned straight lines are simple connectors. The triangular gate
represents negation, while the curved gate represents disjunction.

The requirement represented, recency freshness, says that, assuming that the pairwise
key between member and key server (gcks) used to authenticate the key was not
compromised, if a group member accepts a group key, then the key should not have
expired before the group member requested it (the event “gcks_createkey” describes
the expiration of the old key as well as the creation of the new key). The second
says that, assuming that the pairwise key is not compromised, the member should not
accept a key if it accepted a more recently generated key earlier.
5. Requirements for emerging applications

Most of the protocols that have been analyzed in the literature involve key distribution and authentication. However, there are a number of other applications to which cryptography can be applied that have been emerging in recent years, including electronic commerce, traffic analysis prevention, and denial of service prevention. We take a brief look at each of these below, and then discuss some of the general requirements specification issues that arise.

5.1. Electronic commerce

Electronic commerce protocols must satisfy many of the same properties as key distribution and authentication protocols, but there are some important differences, as follows:

1. Many electronic commerce protocols involve non-safety properties such as fairness, which says essentially that no party should have an advantage over the other one at any point of the protocol. For example, consider a protocol whose goal is that each principal should receive a contract signed by the other. Such a protocol would be considered fair if it there were no way in which a principal could receive a contract without the other principal receiving one. Another example would be a protocol for processing payments. Such a protocol would be considered fair if the there was no way in which the purchasing agent’s account could be debited without it also receiving the goods.

2. The threat model is somewhat more complex than as in Dolev–Yao. Instead of a dishonest intruder pitted against honest principals who follow all the rules of the
protocol, we have semi-honest principals who will not act against their own interest, but may cheat to gain an unfair advantage. However, the model may still include parties trusted to obey the rules, as well as the Dolev–Yao intruder. For example, consider an electronic commerce protocol in which two principals are monitored by a trusted third party. The two principals may try to cheat each other, but they are assumed not to be likely to engage in any activity which will cause them to lose money. The trusted third party, on the other hand, is assumed to follow all the rules of the protocol. The protocol may also be attacked by outsiders, who may be assumed to behave like the Dolev–Yao intruder.

3. The outcomes of the protocols may be quantitative as well as qualitative; for example, one may allow one party to gain advantage over another, but put a limit on the degree of advantage that may be attained. Since electronic commerce protocols usually involve money, it is often straigh-forward to derive such quantitative measures of success.

There is a long history of formulating and characterizing the different properties necessary for electronic commerce protocols, although many of these definitions are tied to particular protocols, and few are formulated with the idea of formal analysis in mind. Markowitch et al. [29] give a survey of the various notions of fairness of exchange protocols, a class of electronic commerce protocols that involves the exchange of goods, information, or commitments. They also provide an informal statement of the desirable properties of an exchange protocol, which we can think of as the exchange protocol’s version of the “secrecy and authentication” formulations of authentication and key distribution protocols in the last section. These defined by Markowitch et al. as follows: viability (it must be possible for the protocol to complete), fairness (the quality of the communication channel being fixed, either all parties attain their goals or none do), and timeliness (the quality of the communication channel being fixed, the parties have the ability to reach a point in which they can abort while preserving fairness). Optional properties include non-repudiability (no party should have the ability to deny having participated in the protocol) and abuse-freeness (no single entity should be able to prove to an outside party that it has the power both to terminate and complete the protocol).

It is not always clear how to formulate such properties so that they are amenable to automated analysis. We consider abuse-freeness as an example, as it has recently received a lot of attention from the formal methods community. The difficulty in the case of abuse-freeness is the notion of being able to prove to an outside party that one has a certain capability; the notion of “proof” is a very general one, and difficult to capture formally. Most attempts at modeling abuse-freeness have either left of the provability condition or modeled it in a very limited sense. For example, Shmatikov and Mitchell [44] and Chadha et al. [11] drop the provability criterion and simply consider the stronger requirement that there be no state in which one of the parties can determine the outcome of the protocol. Buttyán and Hubaux [8] tackle another property similar to but weaker than abuse-freeness: rational exchange, which says that if it party deviates from the protocol, it does not gain any advantage by doing so. They show how this can be naturally formulated in a game-theoretic model. Kremer and Raskin
address the notion of provability, but in a strictly limited sense. They define the ability to prove in terms of the ability to produce a digitally signed message from a principal starting the protocol, and the ability to complete or abort is described in terms of strategies available to a principal. They then come up with a stronger property than abuse-freeness, which says that, at any point in the protocol, it should be impossible for a single principal to display a digitally signed message from another party saying that it started the protocol, unless the other party has a strategy to successfully complete the protocol.

We can see from this brief history that the problem of specifying and characterizing non-safety properties for electronic commerce protocols is even more challenging and varied than the problem of specifying properties of key distribution and authentication protocols, and that there are still a lot of issues to be worked out. For example, it might be possible to get a better approximation of the meaning of the notion of proving to an outside party by using a non-interference-based approach: a principal $A$ is unable to prove to a principal $B$ that an event occurred if, for any set of actions by $A$, the set of all traces visible to $B$ in which the event occurred is the same as the set of all traces visible to $B$ in which the event did not occur.

5.2. Denial of service

Denial of service attacks generally fall into two broad categories: redirection attacks, in which a principal is tricked into believing that a resource is not available, and resource exhaustion attacks, in which a principal is tricked into expending its resources until they are exhausted. Redirection attacks can be prevented by the appropriate use of authentication, and techniques covered in Section 3 should be adequate for specifying their prevention.

Resource exhaustion attacks are more tricky. It is probably impossible to prevent them completely, but, if one can identify the source or sources of the attacks, one can stop the attack by refusing any further communication from those sources. Thus, authentication can be used to curtail resource exhaustion attacks. This can be summed up by a design principle that says that one should delay operations that expend resources until authentication is in place. On the other hand, strong authentication also requires the use of resources, and could itself be exploited in a resource exhaustion attack. This has lead to the design of protocols that use weak but cheap authentication first followed by stronger but more expensive encryption later. Meadows [31] formalizes these two principles in a model based on Gong and Syverson’s fail-stop model, augmented with a cost function that can be used to compare the resources expended by the attacker with the resources expended by a victim. Resistance against denial of service is then defined in terms of an attacker’s inability to make a protocol diverge from normal behavior (as defined in the fail-stop model) without expending substantially more resources than the defender. More recently, LaFrance and Mullins [26] have adapted this approach to modeling resistance to denial of service to a model based on information flow.
5.3. Traffic analysis

The goal of traffic analysis is to find out the source and destination of traffic. In recent years, there has been an increasing amount of work in the design and implementation of anonymizing networks that prevent such analysis, although the history of research in this area goes back to Chaum’s [12] work in the 1980s.

Anonymity is clearly a kind of secrecy, but it is different than secrecy of a key. In the latter case, one is trying to hide the value of a particular piece of data. In the former, one is trying to hide the relationship between two pieces of data: the relationship between the sender (or receiver) of a message, and the message itself. Thus, an attacker might know the contents of a message, and know the identities of all principals present in a network, but not know which principal sent the message or which received it.

The first formalization of anonymity was due to Merritt [34]. This was the notion of hidden automorphisms. Consider a network with a set of principals and an action where one of these principals sends a message. Consider now an automorphism that changes the names of the principals. If an observer’s view of the network is left unchanged by the application of such an automorphism, we can say that observer is ignorant of the sender of the message. More recently, Schneider and Spirodopoulos [41] used a similar definition of anonymity defined on CSP processes to analyze Chaum’s dining cryptographer’s protocol [12]. In this definition, anonymity on a set of events means that events from that set should be indistinguishable to an observer in the sense that if one could have occurred then so could have any. Thus, a set of permutations on an anonymity system would behave like Merritt’s hidden automorphisms.

The above models describe various forms of perfect anonymity. But this is generally too strong a requirement. Realistic systems are generally only able to provide partial anonymity. An intruder who can observe message traffic over time can, in general, piece together some partial information. What is needed is to bound the intruder’s ability to do this, for example, by determining the rate at which it can learn identities of senders and receivers, or the probability of its correctly guessing senders’ and receivers’ identities at some point in time.

There have been several approaches taken to this problem:

1. *Model an intruder of limited ability*: This is what is done by Syverson and Stubblebine in [49]. They note that it may be unrealistic to require an anonymity system to be secure against an omnipresent omniscient intruder. Instead, they posit a group of intruders who may be able to share some information, but not all the time, and will have access to different parts of the network. They describe the different types of assumptions about the ways in which groups of attackers can share information, and develop a calculus for deriving the different types of information the principals can share.

   To consider an example of the type of problems that such a calculus could take, consider two intruders, each of whom is eavesdropping on a different set of locations of a network, trying to determine the sender of a particular message. In the Syverson–Stubblebine calculus, each combination of locating and eavesdropper is
considered a principal, and each intruder may be thought of as a collective group of such principals, such that any information known by one principal is known by the collective. What each collective group knows from its eavesdropping is an or-group of possible senders of the message. The two principals, assuming that they cannot or do not communicate, form an or-group of their own. If they do communicate, they form a larger collective group.

2. Specify the information which must be hidden from an attacker: This approach is taken by Hughes and Shmatikov [23]. They develop a logic for describing the different types of features one might want an anonymizing network to hide from an attacker attempting to perform traffic analysis, such as preventing an attacker from distinguishing individuals in a set of possible senders and receivers of a message, revealing the type but not the identity of a sender or receiver (e.g. merchant or customer in an electronic commerce protocol), revealing the identity of all senders of messages in the system but not revealing which sender sent which message, hiding whether or not two messages have the same sender, and so forth. They show how a number of different types of anonymity can be formalized in their system, including the different guarantees offered by most of the currently available anonymity systems.

3. Specify the intruder's ability to guess information about the sender or receiver of a message: Most actual attacks on anonymity protocols do not give an attacker actual knowledge of the sender or receiver of a message; rather, they increase the chance that an attacker's guess might be correct. Thus, it would be useful to be able to formulate requirements in terms of such probabilistic measures. In general, it is difficult to do this in a way such that meaningful formal analysis is possible, given the discrete nature of most tools. However, probabilistic model checkers are beginning to become available, and in the cases when it is possible to apply them to the problem it makes sense to formulate probabilistic requirements. This is done by Shmatikov in [38,43] in which the protocols and their requirements are modeled in terms of Markov chains, making them amenable to analysis by probabilistic model checkers.

5.4. Incorporating cryptographic notions of correctness

Although the application of formal methods to cryptographic protocol analysis has attracted a lot of attention recently, analytic methods have a longer history. These methods derive from lower level, cryptographic considerations, in which correctness is defined in terms of probability and complexity theory. These models are not, in general, amenable to formal analysis given the capacity of present-day tools, but they provide a sounder mathematical basis for correctness than the standard models used for formal methods, which usually rely unsupported on “black-box” assumptions about the behavior of cryptographic operations.

There is a growing amount of research on unifying the logical models used by formal analysis with the analytic ones used by cryptographers. Some of the earlier work in this area concentrated on developing logical specification languages that took cryptographic considerations into account. The most complete example of this approach is probably the work of Mitchell et al. [36]. More recent work, however, is intended to
have the ultimate effect of simplifying the expression of requirements, since it con-
centrates on using the cryptographic model as a semantics for the Dolev–Yao model, 
thus making it possible to specify requirements at the Dolev–Yao level as long as the 
cryptographic assumptions are met. As an example of this latter approach, consider the 
work of Abadi and Rogaway [1], which considers a complexity-theory-based model 
as a semantics for a logical system, although it restricts itself to secrecy requirements 
and passive attackers. More recently, Backes et al. [3] have been developing versions 
of such standard properties as fairness and liveness, bisimulation, and non-interference 
that are compatible with computational probabilistic notions of cryptographic correctness, and in some cases, have been able to use theorem provers to prove protocols 
correct according to these definitions. They have also begun work on the development 
of a library of cryptographic objects [3] designed so that they can prove that a model 
closely akin to the standard Dolev–Yao model is sound with respect to an implemen-
tation developed using the objects in the library and a polynomially bounded intruder. 
This, again, should allow one to specify security requirements in terms of the standard 
Dolev–Yao model, as long as only objects from the library are used. A similar, but 
somewhat different approach, is taken by Canetti [9] in the development of a set of 
cryptographic primitives that are universally composable, so that security of the cryp-
toalgorithm is guaranteed even when the protocol is running in an arbitrary multi-party 
environment. This property also has the potential to be used to replace the informal 
notion of cryptographic soundness in the standard Dolev–Yao model.

5.5. Emerging trends

We can see a number of trends that emerge from the disparate applications that we 
have presented here. One is that the attacker model becomes richer and more complex. 
Indeed, in several cases we can no longer speak of a single attacker, but a collection 
of attackers with different goals and capabilities. In future work on requirements spec-
ification, it may be as necessary to concentrate as much on specifying the expected 
behavior of the attacker or attackers as it is on the desired behavior of the protocol in 
face of the attackers.

Another issue that comes up, particularly in the specification of requirements for 
electronic commerce protocols, is the tension between secrecy and authentication that 
comes from the different kind of threat model. In the Dolev–Yao model, it is assumed 
that the only goal of the attacker is to subvert the goals of the protocol for the other 
principals, either by discovering secrets or preventing the detection of incorrect protocol 
eexecutions. In the electronic commerce model, there may be a sort of principal of 
intermediate honesty who may try to cheat to achieve its own ends, but who will 
be trusted not to engage in certain types of behavior that is harmful to itself. Thus, 
we may have a situation in which a given principal will not have access to some 
types of information, but will need to see certain other types. For example, in the 
SET protocol, principals must agree on a transaction without necessarily being able 
to see all its components; this greatly complicated the task of specifying the security 
requirements in [42].
An even more outstanding example occurs in the successors to the Internet Key Exchange. Recall that, as Lowe’s analysis showed, the Station-to-Station protocol does not satisfy non-repudiation, in the sense that, if Alice initiates a key exchange with Bob, she can always claim later that she really was attempting to initiate a version of the protocol with someone else. However, for two of the proposed successors to the Internet Key Exchange protocol, JFKr [2] and IKEv2 [24], this lack of non-repudiation becomes a desirable property: “plausible deniability,” which says that an outside party should have no way of determining whether or not Alice actually initiated the protocol with Bob.\(^2\) We can consider plausible deniability to be a type of secrecy property with respect to the outside observer; certain information, that is, the name of the principal with whom Alice is attempting to communicate, is kept secret from the outside observer. Thus, we have two contradictory properties: one, non-repudiation, falling more or less into the class of authentication properties, and the other, plausible deniability, falling more or less into the class of secrecy properties.

Tensions such as the above arise when there are different stakeholders involved with different requirements and expectations and only limited trust in one another. For example, in SET merchants have a requirement for accountability on the part of customers, while customers have a requirement for protection of their private data from merchants. As more and more daily transactions that must satisfy parties with different and often conflicting expectations come to rely on cryptographic protocols, it is likely that this situation will arise more often. Thus, one of the goals of cryptographic protocol analysis, as is the case for other complex systems, will be to determine that the requirements themselves do not contradict each other.

6. Conclusion

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

There are of course a number of areas in which work on cryptographic protocol requirements needs to be extended. One is in making the requirements language user-friendly. Security protocols, and thus their requirements, can be complex; even more so when one must consider operation in partial failure modes such as compromise of temporary session keys. Thus, it makes sense to concentrate on ways of making requirements languages easier to use. In this paper, we discussed some of the work on graphical requirements languages that attempts to address this problem. Of course, graphical requirements languages, and other methods for representing complex requirements, can only take us so far. We expect that one of the emerging problems in this

\(^2\) We do note that in JFKr Bob will be able to tell that Alice initiated the protocol with him, although he will not be able to prove it to anybody.
area will not only expressing complex requirements, but limiting the complexity of such requirements in the first place.

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

Acknowledgements

I would like to thank Iliano Cervesato for providing the graphics for the strand space and the NPATRL specifications, and the referees of this paper for their comments, which much improved the presentation. This work was funded by ONR.

References


