In IEEE Transactions on Software Engineering, 22:1, January 1996, pp. 16-30

A Model for Secure Protocols and Their Compositions

Nevin Heintze and J.D. Tygar, *Member, IEEE*

Abstract—This paper develops a foundation for reasoning about protocol security. We adopt a model-based approach for defining protocol security properties. This allows us to describe security properties in greater detail and precision than previous frameworks. Our model allows us to reason about the security of protocols, and considers issues of beliefs of agents, time, and secrecy. We prove a composition theorem which allows us to state sufficient conditions on two secure protocols A and B such that they may be combined to form a new secure protocol C. Moreover, we give counter-examples to show that when the conditions are not met, the protocol C may not be secure.

Index Terms—Authentication, clocks, communication, composition, computer security, cryptography, formal methods, logic of authentication, model, model checking, protocol analysis, protocols, protocols-composition of, protocol analysis, security, time, timed models.

1 Introduction

What does it mean for a protocol to be secure? How can we reason about secure protocols? If we combine two existing protocols into a common protocol, what can we say about the security of the new protocol?

This paper develops a foundation for reasoning about protocol security. We adopt a model-based approach for defining protocol security properties. This allows us to describe security properties in greater detail and precision than previous frameworks for reasoning about protocol security.

Some of the most advanced work in this area is represented by the Logic of Authentication (developed by Burrows, Abadi, and Needham [11]) which presents a proof-based (or rule-based) approach to reasoning about security properties of protocols. In contrast, we develop a model-based definition of security from first principals. This model provides a comprehensive formal description of the possible actions and interactions among agents, and includes notions of an agent's beliefs and knowledge about messages. (*Messages* include keys, nonces, secrets, text, names, etc.). Given these basic notions, we then define protocol security in term of preservation of properties: first, we can define what it means for a given state to be secure; next, we can reason about protocols that maintain this property.

Our model allows us to reason about time in protocols in a concrete manner. This means that we do not need to rely on a broad notion of "freshness," but that we can define freshness in terms of more primitive concepts.

A highlight of the paper is an account of compositions of protocols. Suppose that we have two protocols A and B which we wish to use together to form a new composite protocol C. For example, A may call B as a subprotocol, or A and B may be concatenated, or A and B may run simultaneously as coprotocols. If A and B are secure, under what conditions is C secure? We state sufficient conditions on A and B guaranteeing the security of the composite protocol C. Moreover, we give counter-examples to show that when the conditions are not met, the protocol C may fail to be secure.

We split the notion of security into two parts: we define the secret-security and the time-security of protocols. By secret-security, we mean that messages believed to be secret are never revealed. By time-security, we mean that stale messages cannot be replayed to subvert the protocol. Although these properties are easy to state informally, a precise account of them must address a number of subtle temporal issues. We consider these properties in a wide framework making them applicable to any reasonable notion of time.

In summary, our paper includes the following novel aspects:

- A detailed development of an elementary model theory of agent interaction that provides an independent definition of security.
- A very general treatment of time (for example, we make no assumptions about global synchronization of time) that supports very general reasoning about interleaving, repeating and composing protocols.
- An explicit account of agent beliefs and nonces, including asynchronous generation and expiry.
- A composition theorem on secure protocols.

2 Models and Security

2.1 Background

Consider a distributed network of communicating agents. Agents can send and receive messages, encrypt and decrypt messages, and generate new messages, keys and nonces. Most communication proceeds according to agreed proto-

- N. Heintze is with AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974. E-mail: nch@research.att.com.
- J.D. Tygar is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: tygar@cs.cmu.edu.

Manuscript received November 1993; revised April 1995.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number S95061.

cols. Such protocols specify how an agent should interpret and respond to messages. The purpose of these protocols may be authentication, key distribution, or information sharing. For example, in the simple protocol given in Fig. 1, an agent A asks a server S to establish an A–B session key (by sending the message $\{A, B\}_{K_{AS}}$), and S responds by sending a secure key K_{AB} to A and B. We use $\{M\}_K$ to denote the encryption of message M with key K, and so $\{A, B, K_{AB}\}_{K_{AS}}$ denotes the message with components A, B and K_{AB} encrypted with K_{AS}. The keys K_{AS} and K_{BS} are assumed to be A's and B's respective secure keys for communicating with the server S. Unfortunately, this protocol is not secure. It is particularly vulnerable to replay attacks: an adversary can record messages 2 and 3 from a valid run of the protocol, and then replay them at any time to convince either A or B to reuse a stale session key.

Message 1	$A \to S$:	$\{A, B\}_{K_{AS}}$
Message 2	$S \to A$:	$\{A, B, K_{AB}\}_{K_{AS}}$
Message 3	$S \to B$:	$\{A, B, K_{AB}\}_{K_{BS}}$

Fig. 1. A simple (and insecure) protocol.

2.2 The Model-Based Approach

One way to reason about the security of protocols is to use models. Models characterize how agents interact. They specify how messages are sent and received, what messages a particular agent can assemble and transmit and which actions an agent can perform at a particular time. In short, models define what can happen.

Given a definition of what can happen, we can study security properties by asking: can anything bad happen? That is, in the space of all possible situations and interactions, is it possible for secrets to be leaked, session keys to be compromised, stale keys to be re-established by replay of old messages, etc.

For example, consider again the protocol in Fig. 1. Fig. 2 presents one possible trace of agent behavior. It records the actions of a server S, agents A and B, and an adversary D. The first part of the trace consists of a run of the protocol, at the end of which agents A and B conclude that K_{AB} is a good session key for A–B communication. In the second part of the trace, an adversary D replays one of the earlier messages to confuse A. At the end of the second part of the trace, A again believes that K_{AB} is a good session key for A–B communication. This puts A and B at considerable risk, because now the adversary D is free to resend old messages using K_{AB}, or worse, if the adversary has broken K_{AB}, it could construct its own messages using K_{AB} and effectively masquerade as B.

This is only one of many possible traces of behavior for A, B, D, and S. Some traces of agent actions are not sensible: for example, a trace in which an agent receives a message before it is sent, or a trace where an agent encrypts and sends a message using a key that it does not know about. We need to have traces correspond to feasible agent interactions. These traces are called models. The definition of what constitutes a model is the central component of a model-based approach to protocol analysis. Such a definition gives rise to a class of potential models which explicitly describe all possible agent behaviors. (There are of course many model definitions, corresponding to different assumptions about agents and communication. Each model definition identifies its own class of models.)

2.3 Overview of Our Model

In our model definition, agents are viewed as nondeterministic machines with some internal state, communicating over public channels by sending and receiving messages. The internal state of an agent determines what action an agent will take next. The behavior of some agents is determined by a protocol. Some agents (such as adversaries) will not adhere to the protocol. They can behave arbitrarily. Other agents may be partially faithful (for example, clients in an electronic commerce protocol may try to cheat). Our model definition views a protocol (and an accompanying specification of the level of faithfulness of each agent) as a constraint on the possible actions of agents. More specifically, each agent is associated with a next-action function that maps its current state into the set of possible next actions. Adversaries are unconstrained, while completely faithful principals (such as servers) are fully constrained by the protocol.

Returning again to the protocol in Fig. 1, observe that the server S should only send the key K_{AB} to A and B if it believes that K_{AB} is a good key for A–B communication. Agent state must therefore include beliefs about what keys are secure. In our model definition, the internal state of each agent consists of three components:

1) the set of messages and keys known to the agent.
2) the set of messages and keys believed by the agent to be secret (and with whom the secrets are shared).
3) the set of nonces recently generated by the agent.

These beliefs do not need to be as general as the beliefs that are typically used in proof systems such as the Logic of Authentication [11].[1] This is because, in our model definition, beliefs are only used to capture a very basic notion of agent state. We do not, for example, need to include complex beliefs about the freshness of compound messages received from other agents. We explain this issue in more detail in the next subsection.

Before we can use our model to answer question such as "does the protocol protects secrets such as session keys or confidential information," we must first determine what are the secrets in the model. This is typically implicit in the protocol specification, and is made explicit in our model by the next-actions function described above. Since most protocols assume that agents possess certain initial shared keys (for example, to talk to a server), we must ensure that models start in some consistent configuration such that the initial keys are indeed secure. In other words, protocol security in our model is about preservation of properties: we define what it means for a given state to be secure and then a secure protocol is one that maintains this property.

1. However, in principle, more complex beliefs could be used as part of agent state.

S's events	A's events	B's events	D's events
	$send\{A, B\}_{K_{AS}}$		
$receive\{A, B\}_{K_{AS}}$			
$send\{A, B, K_{AB}\}_{K_{AS}}$			
$send\{A, B, K_{AB}\}_{K_{BS}}$			
	$receive\{A, B, K_{AB}\}_{K_{AS}}$	$receive\{A, B, K_{AB}\}_{K_{BS}}$	$receive\{A, B, K_{AB}\}_{K_{AS}}$
	conclude: K_{AB} *is good*	*conclude:* K_{AB} *is good*	$save\{A, B, K_{AB}\}_{K_{AS}}$
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮
			$resend\{A, B, K_{AB}\}_{K_{AS}}$
	$receive\{A, B, K_{AB}\}_{K_{AS}}$		
	conclude: K_{AB} *is good*		

Fig. 2. One possible trace of agent events.

One of the more subtle aspects of our model is its treatment of time. We use an abstract, distributed notion of time. There is no global ordering of agent events, only a local ordering of each agent's events. The belief component of an agent's state is continually changing: it evolves asynchronously and nonmonotonically. Beliefs about nonces are established by an explicit nonce generation event (each such event is guaranteed to generated a fresh nonce). Beliefs about secrets (keys or messages) are established either by an explicit message generation event (again such events are guaranteed to generate fresh messages) or by a change of internal state in response, for example, to the receipt of a message. Critical to our model is the idea that nonces and beliefs expire after some (undetermined) time: as part of the definition of model, the following two conditions must hold:

1) Each nonce eventually expires.
2) Each belief of each agent eventually expires (but it may be re-established at some later stage).

3 Proofs versus Models

Model-based approaches focus on agent interaction, and the central question is: what interactions are possible? In contrast, proof-based approaches, such as the Logic of Authentication [11], focus on formulas representing agent beliefs, and the central question is: What beliefs are provable?

3.1 Proof-Based Approaches

Proof-based approaches typically consist of two components: a language for expressing properties about keys, messages and agent beliefs, and a collection of rules for reasoning about these properties. For example, we could write $A \overset{K_{AB}}{\longleftrightarrow} B$ to denote the property that K_{AB} is a secure key for communication between A and B, and the compound formula $A \mid\equiv A \overset{K_{AB}}{\longleftrightarrow} B$ to denote that A *believes* that K_{AB} is a secure key for communication between A and B. Using such formulas, we can write rules such as

$$\frac{A \mid\equiv A \overset{K_{AB}}{\longleftrightarrow} B,\ A \lhd \{X\}_{K_{AB}}}{A \mid\equiv B \mid\sim X}$$

where $A \lhd \{X\}_{K_{AB}}$ denotes that agent A sees the message X encrypted with key K_{AB}, and $A \mid\equiv B \mid\sim X$ denotes that A believes that B once said X. This rule expresses the following claim: if A believes that K_{AB} is a secure key for communicating with B, and A sees some message X encrypted with K_{AB}, then A is entitled to believe that, at some time in the past, B said the message X.

3.2 Comparison with Models

The main advantage of proof-based systems is that they are based on simple, intuitive reasoning rules and are easy to use. Today, comprehensive proof-based systems such as Burrows, Abadi and Needham's Logic of Authentication are the most useful and widely used tools for discovering problems in protocols.

However, proof-based systems gloss over a number of important issues. For example, what do the basic properties really mean? The systems do give them an implicit meaning: a property can be identified with the set of all properties derivable from it. But this is not very satisfactory: it is indirect, gives little intuition, and is dependent on complex interactions among the rules. An even more perplexing question is how to develop and justify the proof rules. These rules are usually carefully constructed to correspond to certain intuitions about how agents interact and how a protocol unfolds. However, these rules often make many implicit assumptions. For example, if we are reasoning about a protocol in which two agents A and B establish a session key K_{AB} for secure A–B communication using a server S, then the example rule given above relies on a number of assumptions, such as

- Agents A, B, and S do not deliberately or mistakenly send the key K_{AB} in plain text.

- Agent S never sends messages encrypted with K_{AB} (note that this could be very subtle, since it could involve an adversary resending messages to try to confuse S).

In summary, an important issue arises with proof-based approaches: if we apply a particular proof-based reasoning system to a protocol, and the system answers "yes, the protocol is secure," then what have we achieved? What has been proved? Can the protocol be broken? One simple answer is: if the proof rules are "correct" then the protocol is good. A more comprehensive response is that if the assumptions on which the rules are based are satisfied, then the protocol is correct. The issue, then, is to isolate the assumptions and determine when they are satisfied. This is very difficult, and has thus far progressed only on a trial and error basis.

In contrast, model-based approaches address the question of meaning in a very direct manner. To define a model one must explicitly identify assumptions. Because of the subtleties involved in reasoning about protocols, such a definition can be surprisingly detailed and complex, particularly if one is to give a satisfactory account of time.

An apparent disadvantage to models is their lack of direct reasoning principles. Model-based approaches define security in terms of properties that hold over all possible traces of agent interactions, so, in principle, we must construct each of these traces in turn and check that the appropriate conditions hold. This is not feasible since there may be an infinite number of different traces and the traces themselves may not be bounded. However, for some restricted classes of protocols, we can limit the traces that must be considered to some finitely representable collection, and thereby develop *model-checking* algorithms. Hence models have two uses: they not only provide a basis for studying the security properties and evaluating proof-based systems, but they can also be used to develop model-checking methods for protocol verification.

3.3 Beliefs

We now describe an important technical difference between our model and the Logic of Authentication. The Logic of Authentication contains an expressive language for representing and reasoning about beliefs. For example, we can express beliefs about the freshness of compound messages received from other agents, beliefs about the beliefs of other agents, as well as beliefs about the security of keys and messages.

In contrast, our model uses a limited form of belief to capture part of the state of an agent. These beliefs are either about the freshness of nonces, or about the security of messages (in particular, which secrets are shared with whom). More complex beliefs are not required because the process of establishing properties of a protocol does not proceed by reasoning about formulas. Rather, the focus is on what interactions can happen. For example, suppose that an agent receives a message M that contains a new key from a server. In the Logic of Authentication, the agent will need to establish that M is fresh before it will conclude that the key is a valid key. If it cannot establish that M is fresh, then it will not be able to conclude that the key is valid. In our approach, the agent will just follow the protocol—if the protocol conditions are satisfied (that is, the message has the right format, etc.), then the agent will assume that the key is valid. If there is a problem with the protocol, and the key is not in fact valid, then the model is not secure.

In other words, when the Logic of Authentication is applied to a faulty protocol, the fault becomes evident when the target belief of the protocol cannot be proven. When our model is applied to a faulty protocol, then some of the models of the protocol will not be secure. The absence of security might arise from an agent belief about a secret that is not valid, or a model that does not satisfy the appropriate belief expiry conditions.

To summarize, the role of beliefs in our model and the Logic of Authentication are different. Beliefs used to describe agent state in our model bear little relationship to the actual protocol properties that we can establish. However, in the Logic of Authentication, the class of properties that we can reason about is exactly the class of formulas used to express beliefs in the logic. (In Section 4.2, we give another illustration of the differing roles of beliefs.)

4 Related Work

4.1 Models

The idea of using models to reason about security has been explored in a number of papers. Early work by Dolev and Yao [12] considers the security of simple *ping-pong* protocols in which agents exchange messages and acknowledgements in the context of a public key cryptosystem. Agent interactions are modeled using strings (sequences) of encryption, decryption, naming, and name-matching operations. The problem of reasoning about these interactions is embedded into a word transformation problem for which a polynomial-time algorithm is presented. Subsequent work has considered more general models addressing a larger class of protocols and a wider variety of security properties. For example, [20] describes a model based on a trace of messages sent and received by each agent in the system, much like the model described in this paper. However, it does not address agent beliefs, and so is not applicable to key exchange protocols. It also does not address issues such as the expiry of nonces. Other relevant works are [16], [19].

4.2 The Abadi-Tuttle Model

Only a few papers have considered security models that deal directly with the issue of time. Abadi and Tuttle [2] use model theoretic constructions to clarify and explain the Logic of Authentication. At a superficial level there are a number of similarities between this model and our model, such as the use of traces. However, the two models differ in fundamental ways.

First, consider the treatment of nonces and beliefs. In Abadi and Tuttle's model, time is global and synchronized. For example, nonce expiry happens at time 0 (the first state of the current epoch): any nonces used before this are considered stale for the current epoch, and nonces used after this are considered fresh throughout the current epoch.

Hence, the possibility of behaviors arising from nonces expiring at different times is not considered.

Second, consider the nature of beliefs. In Abadi and Tuttle, beliefs about shared secrets are essentially static: If a belief about a secret is true at one time, then it is true for all time. In our model, beliefs are continually changing: an agent may believe that a message is secret at one time, but not at some later time (this is an important part of the mechanism for defining time-security). In fact the whole notion of belief differs greatly between the two models. The Abadi and Tuttle model defines what beliefs are true at each point in the model. In our model, beliefs are used to capture one aspect of agent state at each point in the model. The beliefs that an agent has at a particular point may or may not be true (the model makes no judgment about an agent's beliefs on a point by point basis).

Third, the two models have very different structure. The central component of the Abadi and Tuttle model is a specification of the truth of each (Logic of Authentication) formula at each point in the model. This, in turn, is used to justify the proof rules of the logic: it is argued that if the assumptions of a rule are true at a particular point in the model, then the rule's conclusion is also true at that point in the model. As a result, their model closely follows the structure of the underlying logic. In contrast, our model seeks to construct a basic and independent definition of security properties. The basic idea is to build a model that characterizes all potential interactions, and then develop notions of security based on properties that hold over the class of all models. Our model therefore focuses on the mechanics of how agents interact. For example, we have explicit message/nonce generate events and explicit belief expiry events.

Fourth, the most significant difference is that these two models have very different purposes. To illustrate this, consider the belief P *controls* φ in the Logic of Authentication. In the Abadi and Tuttle model, the formula is true if, whenever P sends a message[2] containing φ, it is the case that φ is true at time 0 and at all times after this. Such a belief is the principal way to indicate the trustworthiness of a server (or other key generating agent) in the Logic of Authentication. It is, however, a very strong condition. For example, if φ states a property such as M is secret, then P *controls* φ means that P is trusted never to reveal the secret to an adversary, and any agents that P shares the secret with (and any agents that they in turn share the secret with) are similarly trusted.

In contrast, our model does not have any notion of P *controls* φ. Instead, the philosophy of our model is: let's see what happens when the protocol is run (and rerun), and see if all reachable states are secure. This distinction underlines the central difference between model-based and proof based approaches to protocol security.

4.3 Other Related Work

A number of other papers combine proof and model approaches. Building on the Abadi and Tuttle model, [24] extends the Logic of Authentication to include temporal operators, and thereby provide a more expressive treatment of time. [4] uses a notion of agent histories (that record messages sent and received by each agent) to develop a temporal logic for reasoning about security. It does not address a number of key issues relating to time such as the expiry of nonces and beliefs. [22] use a logic language equipped with temporal operators for specifying protocol requirements to be checked by the NRL Protocol Analyzer [19].

In general, time has received increasing attention in the literature on protocol security analysis. The Logic of Authentication [11] was one of the first papers to directly consider time, and provide reasoning principles for reasoning about the freshness of nonces and messages. However, the notion of time was somewhat limited; protocols were static in the sense that one could only consider a single run of a protocol. The idea of considering instances of protocols and potential interactions among them was subsequently considered in [5].

5 Messages, Keys, and Encryption

We begin by considering the basic notions of messages and encryption. In this paper we consider security properties that are independent of the underlying message representation and encryption scheme. We assume an idealized model of message representation and encryption. Specifically, we assume that messages are independent; that is, having one message does not give any information about another message. Second, we assume that the only way to obtain any information from an encrypted message is to have the appropriate key. This is often referred to as "perfect encryption" in the literature [6], [14].

Of course, in general, there may be important interactions between a protocol and the underlying encryption scheme used in its implementation. In particular, if an encryption scheme is amenable to a probabilistic analysis, then it would be desirable to extend the analysis to any protocols using the encryption scheme. However, such an analysis of a protocol is likely to be dependent on the specific properties of the encryption scheme, and hence must be done on a case by case basis.

In contrast, the perfect encryption assumption leads to a more abstract notion of protocol security. Specifically, it yields a lower bound analysis: if a protocol is not secure under the perfect encryption assumption, then it fail to be secure regardless of the encryption scheme used. Another view of the perfect encryption assumption is that it provides a way to decompose the problem of analyzing a protocol into two subproblems: 1) an analysis of the protocol that focuses on the intrinsic security properties of the protocol by ignoring details of the encryption scheme used, and 2) an analysis of how the encryption scheme interacts with the protocol. This paper is concerned with the former problem.

To formalize the perfect encryption assumption, first let $\mathcal{B}$ denote a set of *basic messages*, which shall represent the keys, the nonces, and the nondecomposable[3] message of the model.[4] Notationally, we shall use K to denote elements of $\mathcal{B}$

2. This does not include the case when P just forwards a message.

3. That is, messages that are not encryptions or compositions of other messages.

4. We will assume that each agent A has a publicly known name (also denoted A) that is a basic message.

that are used as keys, and N to denote those used as nonces. Messages are defined to be the result of composing and encrypting basic messages. Specifically, a *message* M is either

- a basic message;
- $\{M\}_K$, the *encryption* of M with K, or
- $(M_1, \ldots, M_n)$, the *composition* of $M_1, \ldots, M_n$.

where $M, M_1, \ldots, M_n$ are messages, $n \geq 2$, and $K \in \mathcal{B}$ is a key. Now, given a set of message S, an agent has a limited ability to decrypt and decompose these messages into their constituents and also built up new messages by encryption and composition. Specifically, define S^*, the set of messages *deducible* from S, to be either 1) any element of S, 2)[5] M such that $\{M\}_K$ and K are in S^*, 3) M_i such that $(M_1, \ldots, M_n)$ is in S^*, 4) $\{M\}_K$ such that M and K are in S^*, and 5) $(M_1, \ldots, M_k)$ such that each M_i is in S^*.

The construction presented here can also be viewed as a free algebra construction over the generators $\mathcal{B}$, tupling and encryption, with auxiliary operations, call them *decrypt* and *decompose*$_i$, $i \geq 1$, that satisfy the following equations:

$$decompose_i(M_1, \ldots, M_n) = M_i, \quad 1 \leq i \leq n$$
$$decrypt\ (K, \{M\}_K) = M.$$

6 Traces and Some Basic Assumptions

Let $\mathcal{A}$ be the (finite) set of all agents of interest (that is, $\mathcal{A}$ includes not only the principals of the protocol at hand, but also any adversaries). We view each agent as an automaton, with a notion of "state" (including information about beliefs and nonces), and a set of legal transitions (for the principals of a protocol, these transitions must be in accordance with the protocol; for adversaries, the possible transitions are more permissive).

As a first step towards formalizing such a view of agents, we define a *trace*, which is a record of agent interactions (messages sent and received) as well as other agent actions such as the generation and expiry of nonces, keys and secrets. These basic interactions are modeled by *events*, which take one of the following forms: *send*(M), indicating that message M is sent by an agent; *receive*(M), indicating that message M is received by an agent; *generate*(M), indicating that the basic message M is generated by an agent (M may be either a nonce or a key); and *expire*(M), indicating that the message M has become stale. Events can be subscripted as necessary to indicate different occurrences of the same event; for example, if an agent resends a message.

Intuitively, a trace is a specification of a set of events for each agent A in $\mathcal{A}$. However, we have omitted a crucial element—time. The simplest way of introducing time is to associate a time value (for example, a real number) with each event. Unfortunately, such an approach assumes the existence of a global clock. In a networked environment, such an assumption is at best restrictive, and at worst unrealistic. Instead, we shall use a more abstract time that is compatible with systems of unsynchronized clocks, as well as nonstandard notions of time such as Lamport clocks [17].

5. For a public key system, this clause would be modified. Specifically, where K^c denotes the key which is the complement of K, we would have: M such that $\{M\}_K$ and K^c are in S^*.

This is based on a very minimal assumption about time: Each agent has a *local* notion of "before" and "after" that gives a total order on each agent's events. Then:

DEFINITION 1. (Traces) *A trace T is an $\mathcal{A}$ indexed collection of sets of events such that each set is equipped with a total order.*

T_A denotes the set corresponding to the agent A, and T denotes the union of all of the T_A. The total order on T_A is denoted by $<_A$; if $e <_A e'$ then we say that *e precedes e'*. We write $e \leq_A e'$ if either $e <_A e'$ or $e = e'$. For any event $e \in T_A$, we write *succ*(e) to denote the *successor* of e if it exists; that is *succ*(e) is the earliest event in $\{e' : e <_A e'\}$. Note that a trace says nothing about the ordering of events between agents.

The above definition does not admit traces in which two events happen simultaneously. Such a possibility could be accommodated by replacing the notion of total order by the more general notion of a total preorder. However, the extra structure afforded by simultaneous events is inconsequential for the security properties identified in this paper, and so, for simplicity, we shall use total orders. (This bears a superficial resemblance to the treatment of time in the model theory of temporal logic [1], [3], [10].)

An agent typically sends a message to some designated recipient (such information may be implicit in a protocol, or explicit in the plain text or encrypted parts of messages). However, we do not assume that the communication medium is secure, and so we do not guarantee that a message is received only by its intended receiver. Rather, we treat messages sends as broadcasts to the world; any agent can receive any message that is sent. Further, we make no assumptions about the correct behavior of the network—messages may be completely lost, received by only some agents, or even duplicated due to network failures and errors.

The above sketch of the definition of trace is overly permissive: It admits traces of agent interactions that are not physically possible. We now address three aspects of this issue.

First, between any two events there lie only a finite number of events. This can be justified on physical grounds; it is also necessary for technical reasons.

DEFINITION 2. (Bounded) *A trace T is bounded if, for all agents A and all pairs of events e_1 and e_2 in T_A, the set $\{e : e_1 <_A e <_A e_2\}$ is finite.*

Second, it is clear that if a message M is received by an agent, then some agent must have previously sent M. Clearly there must be some consistent way of interleaving the message traces from the various agents such that each message receive is preceded by a message send (this is identical to a requirement used by Merritt [20]). Moreover, we require that this interleaving must be fair. This leads to the following definition:

DEFINITION 3. (Serializable Traces) *A trace T is serializable if there is a total order $<$ on T such that*

1) *the ordering $<$ conservatively extends the orderings $<_A$, $A \in \mathcal{A}$, in the sense that if $e <_A e'$ then $e < e'$,*
2) *each event receive(M) in T is preceded by an event send(M), and*
3) *for all pairs of events e_1 and e_2, the set $\{e : e_1 < e < e_2\}$ is finite.*

Fairness is expressed by part 3) of this definition. We remark that serializability is only a consistency requirement. It does not imply a global ordering of events. In general there will be many such orderings $<$, and so subsequent definitions cannot be expressed in terms of a fixed event ordering. We cannot, for example, state a condition that beliefs about a particular secret held by different agents expire simultaneously.

Third, we require that each event of the form *generate*(M) creates a new basic message.

DEFINITION 4. (Message Generation) *A trace T respects message generation if, for all distinct events of the form generate*(M_1) *and generate*(M_2), *the messages* M_1 *and* M_2 *are distinct.*

In what follows, we use the term trace to mean a trace that is serializable, bounded, and respects message generation.

7 Protocols and the Untimed Model

The definition of trace (and its associated assumptions) captures a notion of agent interaction in which agents are completely free to send and receive messages. This notion does not take into account the constraints on interaction imposed by protocols. A protocol typically identifies a subset of agents (the *principals* of the protocol), and specifies how these agents should interact with other agents. Such a specification is usually described as a sequence of message sends and receives. For example, Fig. 3 presents a variant of the Otway-Rees protocol [21]. Although this description gives explicit information about the form of messages to be sent, a number of side conditions are either implicit or unspecified. First, it is implicitly specified that the nonces used must be fresh. Second, it is implicitly specified that the keys used are appropriate secret keys. Third, the protocol is a collection of rules describing how an agent should send and respond to messages; that is, the protocol is not just the sequence of four messages, but rather a specification that:

- to establish a session key with B, A should send the first message of the protocol;
- if B receives this message, then B should respond with the second message;
- if the server S receives the second message, then S should send out a new secure key using the third message;
- if B receives the third message, then B should accept the new key as a secure key for A-B conversations and also respond with the fourth message, and
- if A receives the fourth message, then the protocol is complete and A should use the new key for conversations with B.

Moreover, the variables N_A, N_B, K_A, K_B, ... are really *parameters* ranging over nonces and keys etc. That is, a protocol is really a template for "transactions" between agents. In general, this template will be invoked many times. It is therefore important to model multiple (and possibly interleaved) invocations of the protocol; each instance will typically involve different values for the parameters.

In summary, a protocol specifies what an agent should do next, based on 1) what the agent currently believes (about keys and nonces) and 2) an event (such as the receipt of a message).

$$\begin{aligned}
A \to B&: \; A, B, \{N_A, B\}_{K_A}\\
B \to S&: \; \{N_A, B\}_{K_A}, \{N_B, A\}_{K_B}\\
S \to B&: \; \{N_A, B, K_{AB}\}_{K_A}, \{N_B, A, K_{AB}\}_{K_B}\\
B \to A&: \; \{N_A, B, K_{AB}\}_{K_A}
\end{aligned}$$

Fig. 3. A variant of the Otway-Rees protocol.

We first formalize the notion of beliefs used in our model. We consider two kinds of beliefs: beliefs about secrecy and beliefs about freshness. Compared to a logic such as the Logic of Authentication, these beliefs are very limited. In principle, more complex beliefs could be introduced. However, this is not necessary, since the beliefs used here are only used to capture a very basic notion of agent state. See Section 3.3 for a discussion of this issue.

A *belief* is of the form *shared*(S, M) where S is a set of agents and M is a message, or of the form *fresh*(M) where M is a basic message. Informally, *shared*(S, M) denotes the belief that only the agents in S can know about M, and *fresh*(M) denotes the belief that M is a recently generated nonce.

An agent state, STATE, consists of a set of beliefs and events (intuitively, this represents an agent's current set of beliefs, and the events that it has experienced). Agent beliefs are included in agent state to capture an agent's view of how a protocol (or protocols) is unfolding: What secrets are shared, which keys are secure, etc. The fact that an agent holds a particular belief at a particular time is no indication that the belief is in fact true. We return to the issue of validity of beliefs later in this section.

Given a state, we can define the set of messages known by an agent. From this set an agent can send messages and construct new beliefs. Define *known*(STATE), the messages constructible from the information present in STATE, to be the following set:

$$\left(\left\{\begin{array}{l}\textsc{state}\text{ contains either}\\ M\text{: } \mathit{fresh}(M),\ \mathit{shared}(S,\ M),\\ \mathit{generate}(M),\text{ or } \mathit{receive}(M)\end{array}\right\} \cup \mathcal{A}\right)^*$$

That is, *known*(STATE) consists of messages that appear in beliefs, messages that have been received or generated, messages used for naming agents, and all messages derivable from the above messages.

Given an agent A, a set of beliefs and a set of events, a protocol specifies a set of possible actions for A. Such actions include sending messages and adding new beliefs to A's current beliefs (for example, adding beliefs about newly established keys). More formally:

DEFINITION 5. (Protocol) *A protocol is a pair* (P, δ) *where* $P \subseteq \mathcal{A}$ *is a set of agents indicating the principals of the protocol, and* δ *is a collection of functions* (δ_A, $A \in P$). *Each function* δ_A *maps any set consisting of receive events, send events, and beliefs (these represent the current state of the principal A) into a set of allowed actions for A. Specifically, if* STATE *is the set input to* δ_A, *then* δ_A (STATE) *is a set consisting of the following two kinds of elements:*

1) *beliefs of the form shared(S, M) where* $M \in$ *known*(STATE), *or*
2) *events of the form send(M) where* $M \in$ *known*(STATE).

Elements of the form 1) in δ_A(STATE) indicate new beliefs that A should add to its set of current beliefs. Elements of the form 2) indicate messages that A should send. We assume that each function δ_A is monotonic. Note that we do not include generate and expiry messages in the notion of state used in the above definition. The model defines the meaning of these events, independent of the protocol.

A trace is *faithful* to a protocol if each principal behaves in accordance with the protocol. The behavior of an agent includes message sends as well as the way an agent manages its beliefs. The management of beliefs not only includes the establishment of new beliefs (as specified by the protocol), but also the expiry of "stale" beliefs (as indicated by events of the form *expire*(*M*) in the trace). To formalize this, we must track the beliefs of an agent at each "point in time." In general, an agent's beliefs are essentially an arbitrary function of time. However, we wish to avoid introducing an explicit notion of time. To maintain this level of abstraction, we observe that it is only necessary to know an agent's beliefs at each event (we shall return to this point later). Hence, in the context of a protocol (P, δ), we define:

DEFINITION 6. A *belief function for a trace T is a mapping from T into finite sets of beliefs. If* $e \in T_A$, *then beliefs(e) is the (finite) set of beliefs held by agent A at event e.*

A model consists of a trace and a corresponding belief function, subject to a number of constraints. Before giving these, we need some additional notation. In the context of a belief function *belief* and some event e in T_A, define *state*(*e*), the state of agent A at event e, to consist of the beliefs in *beliefs*(*e*) as well as the send and receive events that appear in $\{e' \in T_A : e' <_A e\}$. Define BELIEFS_M, the set of beliefs involving M, to be $\{fresh(M)\} \cup \{shared(S, M) : S \subseteq \mathcal{A}\}$.

DEFINITION 7. (Model) *A* <u>*model*</u> *for a protocol (P, δ) is a pair (T, beliefs) where T is a trace and beliefs is a belief function for T. Moreover, for each agent A and each event* $e \in T_A$, *if* $A \in P$ *then:*

1) *If e is send(M) then* $e \in \delta_A(state(e))$.
2) *If e is expire(M) then*
   - $beliefs(succ(e)) = beliefs(e) - \text{BELIEFS}_M$.
3) *If e is generate(M) then*
   - $beliefs(e) \subseteq beliefs(succ(e))$, *and*
   - $beliefs(succ(e)) - beliefs(e) \subseteq \{fresh(M), shared(S, M)\}$, *for some set* $S \subseteq \mathcal{A}$.
4) *If e is send(M) or receive(M) then*
   - $beliefs(e) \subseteq beliefs(succ(e))$, *and*
   - $beliefs(succ(e)) - beliefs(e) \subseteq \delta_A(state(e) \cup \{e\})$.

and if $A \notin P$ *then:*

5) $beliefs(succ(e)) \subseteq known(state(e) \cup \{e\})$.
6) *If e is send(M) then* $M \in known(state(e))$.

In items 2), 3), 4), and 5), if *succ*(*e*) does not exist, then the condition is vacuously true. Items 1) through 4) constrain the behavior of protocol principals; items 5) and 6) constrain the behavior of adversaries. We discuss each collection in turn.

Item 1) states that the messages sent by a protocol principal accord with the protocol. In conjunction with Definition 5, part 2) (the definition of protocol), it also implies that principals can only send messages that they know about. Items 2) to 4) define the effect of send, receive, expiry and generation events on agent beliefs. These conditions have three key consequences. First, deletion of beliefs must correspond to an expiry message. Second, the only possible effects of a generation event are that beliefs of the form *fresh*(*M*) or *shared*(*S*, *M*) are added to the agent's belief set. Moreover, only one belief of the form *shared*(*S*, *M*) may be added. Third, message send and receive events can only add new beliefs, and these must accord with the protocol. Note that in 4), new beliefs added at *succ*(*e*) depend on beliefs at event *e* and also on all events preceding *succ*(*e*) (this includes event *e*). Hence new beliefs are bounded by $\delta_A(state(e) \cup \{e\})$. In a particular model, an agent is not forced to do every action outlined by the protocol. Rather, the protocol constrains what an agent can do. Within this boundary, an agent is free to choose what actions it will perform—this choice is typically dictated by factors such as agent workload, the need to communicate, how recently a message has been sent (for example, at some stage a message may have to be resent because it has been lost).

Items 5) to 6) address nonprincipals. Item 5) states that the beliefs of an adversary can grow only by either message generation or by the receipt of messages. Item 6) states that an adversary can only send messages that it knows about. In effect these two constraints ensure that adversaries cannot randomly guess secrets: they can only learn about secrets by interactions with other agents. We remark that 5) and 6) in fact hold for all agents: For protocol principals, the conditions expressed by 5) and 6) are consequences of the definition of protocol and items 1) to 4).

We now address the meaning of beliefs. Thus far we have essentially used beliefs as tokens to capture part of an agent's state, independent of what the tokens actually mean. These tokens have played an important part in formalizing the mechanics of our model, and in particular, how an agent carries out a protocol. Now, to talk about security of a protocol, we want check whether an agent's secrets and session keys are secure in protocol models. We therefore need to say what it means for particular beliefs to hold in the model. We also need to identify which beliefs must hold. In our model, we shall require that all beliefs of protocol principals must hold.

We begin by defining what it means for beliefs to hold at a particular event. One important part of the behavior of beliefs is that they eventually expire (this allows us to reason about what happens when stale messages are replayed). This issue is addressed in detail in the next section. For now, we shall just consider the time-invariant part of the meaning of beliefs. First consider beliefs of the form *fresh*(*M*). Such beliefs are used to track the behavior of nonces: they are generated when a nonce is created, and apart from that all we require is that they eventually expire. We consider the issue of expiry in the next section; in this section, beliefs of the form *fresh*(*M*) are simply considered to be *true*. Second, consider beliefs of the form *shared*(*S*, *M*). The meaning of these beliefs depends on which agents know about M. That is, a belief *shared*(S, M) is *true* if $M \in$ *known*$_A(e)$ implies that $A \in S$. To summarize, we define:

DEFINITION 8. (Valid Beliefs). *Let (T, beliefs) be a model for protocol (P, δ). A belief b is valid at some event $e \in T_A$ if it is either of the form fresh(M), or of the form shared(S, M) such that if $M \in known_A(e)$ then $A \in S$.*

This defines a notion of pointwise validity of beliefs, and provides a basic element of our definition of protocol security. However, to obtain a meaningful notion of protocol security, we cannot just define that all beliefs must be valid at all events in all models for the protocol. This is inappropriate for a number of reasons. First, we are only interested in the beliefs of principals, since a protocol does not give any assurances about what happens to nonprincipals. Second, no protocol can provide any guarantees about security when security has already been compromised. Instead, we shall define protocol security in terms of *preservation* of properties: a protocol shall be considered secure if, whenever it started in an "initially" secure configuration, all subsequent behavior is secure. To formalize this, we first define what it means for the model to be secure at some "time." This is achieved using snapshots, which are cross-sections of the model.

DEFINITION 9. (Snapshots) *In the context of a model (T, beliefs), a snapshot s is a subset of T that contains exactly one event (denoted s(A)) from each set T_A. A snapshot s is secure if, for all principals A of P, each element of beliefs(s(A)) is valid at each event in s.*

Next we formalize what it means for a model to be initialized in a secure state. In the context of a model M, a belief b is an *initial belief* if, for some agent A and event $e \in T_A$, it is the case that $b \in$ *beliefs*(e), and for each event $e' <_A e$, it is also the case that $b \in$ *beliefs*(e'). Intuitively, an initial belief is one that an agent is given at the start of time, as opposed to a belief that is established by some action of the protocol. Initial beliefs might include the initial keys for agent-server communication. The notion of a model being started in an initially secure state is realized by placing properties on the set of initial beliefs. Clearly initial beliefs must be "initially valid." Furthermore, they must also be consistent in the sense that beliefs about a message must not conflict. For example, if for some message M, an agent A_1 has the initial belief *shared*$(\{A_1, B_1\}, M)$ and the agent A_2 has the initial belief *shared*$(\{A_2, B_2\}, M)$, then problems will arise when A_1 shares M with B_1 and A_2 shares M with B_2. In essence, we prevent this situation by requiring that there is at most one kind of initial belief about each message. However, in the case of beliefs involving messages made up from other messages, this requirement is problematic. We therefore require that initial beliefs not contain other messages; that is, initial beliefs may only involve basic messages.

DEFINITION 10. (Initially-Secure Models) *A model (T, beliefs) is initially-secure if* 1) *there is a snapshot s such that all initial beliefs of principals are valid at all events in $\{e: e \leq e'$ for some $e' \in s\}$ and A in $\mathcal{A}$; and* 2) *all initial beliefs are of the form shared(S, M) or fresh(M) where M is a basic message that is not generated by some event in T, and* 3) *if shared(S_1, M) and shared(S_2, M) are both initial beliefs then $S_1 = S_2$.*

The notion of "initially-secure" is defined not by considering security at a specific snapshot, but rather security at events in and preceding the snapshot, because there may be messages in transit that are not represented by a single snapshot. We can define protocol security in terms of the preservation of security:

DEFINITION 11. (Secret-Security) *A model (T, beliefs) is secret-secure if, for all principals A of P and for all $e \in T_A$, each element of beliefs(e) is valid at each event in T. A protocol (P, δ) is secret-secure if all initially-secure models for (P, δ) are secret-secure.*

This definition provides a rudimentary notion of protocol security: in essence, a protocol is secure according to this definition if agents do not reveal secrets. In the next section, we enrich this definition with a mechanism for reasoning about the freshness of nonces. In doing so, we obtain a definition of protocol security supporting reasoning about security attacks based on stale message replay.

We discuss the definitions given thus far. First, consider the definition of traces. In general, the sets T_A that make up a trace T are unbounded in both directions. That is, there is no requirement for an agent to have a "start" or "end" event. In many contexts however, a protocol is started in some initial state. In this case, it is reasonable to restrict traces so that each set T_A has an *initial event* that precedes all other events. Models that are constructed from such traces shall be called *directed models*. This subclass of models gives rise to a modified definition of secret-security:

DEFINITION 12. (Directed Secret-Security) *A protocol is directed secret-secure if all initially-secure directed models for the protocol are secret-secure.*

This alternative definition turns out to be strictly weaker[6] than the previous definition (Definition 11). It also has certain technical advantages, which shall be employed when we consider composition of protocols.

Second, consider belief functions. Their purpose is to capture the beliefs of each agent at each point in time. However, instead of using a concrete notion of time, we chose to just describe the beliefs at each event. In essence, we use events themselves to reason about time. We now argue that this results in no loss of generality. The main simplification afforded by the use of events to index an agent's belief sets is that all information about what happens to an agent's beliefs between two events is ignored. Now, such information does not affect what messages an agent can send, because the only beliefs relevant to message sending are those current at the time of the send event. Hence, the only possible impact of the new structure is that there may be new beliefs that appear after one event and disappear before the next event. However, this cannot happen, because the only mechanism for an agent to drop a belief is via an event of the form *expire*(M).

Third, consider the behavior of beliefs about shared keys. Suppose that a belief *shared*(S, M) is held by a principal A at some event e in some initially secure model of a secret-secure protocol. By definition, this implies that *shared*(S, M)

6. However, the two definitions coincide for most protocols arising in practice.

is valid at all events in the model. Hence, given any initially-secure model of a secret-secure protocol, all beliefs held by principals of the protocol are true at all events in the model. In other words, beliefs of principals are universally valid.

One implication of this property is that if a principal holds a belief *shared*(*S*, *M*) at some event in the model, then only agents in *S* can ever see the message *M*. In other words, when holding a belief about secrecy, an agent must know all of the agents that will share the secret. While this seems like a useful property to require, it is arguably restrictive. For example, it is not possible to add a mechanism to the model that allows an agent to hold a belief of the form *shared*({*A*}, *M*) and then share the secret *M* with another agent *B* and update its belief to *shared*({*A*, *B*}, *M*). We remark that the main feature of our model that is responsible for the "universal validity" property is the abstract treatment of time. In particular, no assumptions are made about synchronization between agents.

We conclude this section with some general comments about our treatment of beliefs. Recall that one component of our model is the specification of the sets of beliefs for each event at each agent via belief functions. This captures the evolving nature of beliefs during (possibly interleaved) executions of a protocol, and is necessary to account for the behavior of agents. For example, in a protocol such as the Otway-Rees protocol (Fig. 3), agents *A* and *B* participate only if they believe that their initial keys for server communication are secure. More correctly, they will only participate using server-keys that they believe are secure. Hence, beliefs are viewed as an integral part of an agent's state, quite independent of whether or not the beliefs actually hold. In this view, beliefs are part of the mechanism of the model and are internal to it.

Beliefs also have a status as statements which are either true or false at some particular event or snapshot in a model. In this view, beliefs are external to the model and they derive their meaning from the model. So, beliefs have both an "internal" and "external" role to play. This does not involve any inconsistency or mutual dependency because in the internal role, beliefs are manipulated as "data," independent of the truth-meaning given to them in their external role. To illustrate this point, note that adversaries may have beliefs, but we ignore the validity of these: our model only requires that protocols respect and preserve the beliefs of the principals.

8 BELIEFS AND TIME

The treatment of time is one of the most important aspects of the analysis of a protocol. Typically the security properties of a protocol rely on the generation and expiration of nonces to ensure that certain information is fresh. Further, it is expected that propositions such as "this key is a secure key" do not hold forever, but expire when the key becomes stale.

Consider the behavior of nonces. Once generated, nonces typically have some predetermined finite lifetime. For example, a finite life δ may be specified so that if a nonce is generated at some time t, then it is only fresh until time $t + \delta$. Of course, δ may vary from nonce to nonce. Instead of committing to a specific mechanism, we employ a simple and abstract characterization: each nonce eventually expires. This is formalized by requiring that if an agent holds a belief of the form *fresh*(*M*) at some event, then there exists a later event of the form *expire*(*M*). Since the only mechanism for an agent to add *fresh*(*M*) to its set of beliefs is via an event of the form *generate*(*M*), and each such event is guaranteed to generate a new basic message, it follows that each nonce eventually expires and is never again considered fresh.

Now consider beliefs held by agents. Again, they have a limited life. One difference between nonces and beliefs is that a nonce is believed to be fresh simply on the basis of when it was generated. On the other hand, beliefs are established on the basis of the protocol and the messages that have been received. This means that a belief may be established at some time, considered to be stale at some later time, and then be reestablished at yet another later time. An appropriate expiry condition for beliefs is: if an agent holds a belief of the form *shared*(*S*, *M*) at some event, then there exists a later event of the form *expire*(*M*). In particular, this implies that the only way for an agent to maintain a belief indefinitely is for the belief to be enabled indefinitely. These ideas lead to the following definition:

DEFINITION 13. (Time-Secure) *A model for a protocol P is a* <u>*timed model*</u> *if for each principal A of P and event* $e \in T_A$, *if beliefs*(*e*) *contains a belief of the form shared*(*S*, *M*) *or fresh*(*M*), *then there exists an event* $e' > e$ *such that e is expire*(*M*). *A timed model is* <u>*time-secure*</u> *if, for all principals A of P, if b is a belief that is held by A at some event, then there is an event* $e \in T_A$ *such that b is not held at any event following e. A protocol P is* <u>*time-secure*</u> *if each initially secure timed model for P is time-secure.*

This definition does not distinguish between the expiry requirements of different kinds of beliefs. In particular, different keys may have very different lifetimes. For example, agent-agent session keys have only a short lifetime compared to server-agent keys. To investigate the impact of this distinction on security in our model we can do the following. First, we give a classification of beliefs into short-term and long-term beliefs. Then, we view long-term beliefs as having an infinite lifetime compared to short-term beliefs. This can be achieved by adapting the above definition so that only the short-term beliefs are required to expire. The result is a definition of time-security that is relative to a particular distinction between short-term and long-term beliefs. To analyze a protocol, it may be appropriate to consider time-security properties with respect to a number of different short-term/long-term distinctions. Note that the two limiting cases of the short-term/long-term distinction are secret-security (when all beliefs are long-term) and time-security (when all beliefs are short-term).

The secret-security definition imposes an abstract requirement for belief expiry. In particular, there is no notion of scheduling or ordering of belief expiry: later beliefs can expire before earlier ones, and beliefs established using secure keys can outlive belief in the security of the key. The intention is that our model includes all potential behaviors of a system. (A specific system may of course enforce a particular strategy for constraining the way beliefs expire. In the unlikely event that security properties critically depend on such specifics, one could modify our model to take them into account.)

Why does the definition of time-secure capture an important notion of security? One protocol failure mode is where an adversary replays sequences of old messages to attempt to convince a principal of the validity of a belief. However, in certain circumstances, this may be secure. For example, suppose an agent A sends a message M to another agent B that is intercepted by an adversary Z and does not reach B. Soon afterwards, Z resends M to B. Now, from B's point of view, there is no essential difference betrween this situation and a situation where network latency is abnormally high. In other words, that fact the Z was able to replay a previous message to convince B of a certain belief b was not significant in this case.

As another example, consider a modification of the above scenario. Suppose this time that B receives the message the first time and then believes b. At some later time B then discards the belief b. Then suppose that Z resends M, and on receiving M, B re-establishes its belief in b. (This may happen for example if the time-out interval for the belief b was significantly shorter than the time-out interval for nonces). This situation does not differ in an essential way from the situation where network problems result in two copies of M being received by B, one significantly before the other, and so again this does not indicate a protocol security problem.

Contrast these two examples with the situation where Z is able to replay M indefinitely to convince B of b (as would be the case if M did not contain any nonces). It is exactly the distinction between these two kinds of behavior that the above definition of security is seeking.

Protocol security can now be defined by combining the definitions of secret-secure and time-secure:

DEFINITION 14. (Protocol Security) *A protocol is* <u>*secure*</u> *if it is both secret-secure and time-secure.*

We provide some informal justification for our use of a very abstract notion of time: Clearly, from the point of view of generality, it is desirable to avoid including a specific time framework in the definition model. However in doing so, we must address the issue of whether the resulting definition is generally applicable, because the notion of security may be depend on the notion of time. In other words, we must consider the relationships between an appropriate notion of security in the context of a specific time framework and the definition of security obtained by our more abstract definition of model.

Our definition of security is essentially equivalent to any reasonable definition in the context of a specific notion of time. In other words, our definition captures the essence of what it means for a protocol to be secure. In Appendix A, we provide evidence for this assertion by comparing three models with different notions of time: one is an early version of the model presented in this paper, and the other two differ only in that they incorporated specific notions of time.

We conclude this section by proving that a number of security properties are undecidable. We conjecture that this result extends to cover all models with similar features to our model. However, it appears likely that there are interesting subclasses of protocols that can be defined by syntactic restrictions on the definition of protocol, for which security is decidable.

THEOREM 1. *Given a protocol P, the following questions are undecidable:*

1) *Is P secret-secure?*
2) *Is P time-secure?*

PROOF SKETCH. We have defined protocols as transition systems, and as a result it is straightforward to express Post's correspondence problem [18] as a security question. In particular, it is easy to code any instance of Post's correspondence problem as a protocol such that a belief of the form *shared*($\{A\}$, M) can be held by an agent A if and only if there is a solution to the correspondence problem. Moreover, we can arrange for M to be some message which is not secret (that is, M may be know to agents other than A). Hence the protocol is secret-secure if and only if the correspondence problem has a solution. This outline proves 1). The remaining parts can be proved by similar methods. □

We remark the above result exploits the generality of the definition of protocol and the inherent power of the tupling operation, but makes only minimal use of encryption. In contrast, previous undecidability results for protocol security use complex algebraic properties of richer encryption systems. For example the proof of undecidability for *half-word ping-pong* protocols [13] relies on the structure of a more general encryption-decryption scheme that allows commutativity of some pairs of operators.

9 Composition of Protocols

Consider the problem of combining protocols. That is given protocols $p_1 = (P_1, \delta^1)$ and $p_2 = (P_2, \delta^2)$, we wish to obtain their composition $p_1 \cup p_2$ defined by $(P_1 \cup P_2, \delta^1 \cup \delta^2)$, where $\delta^1 \cup \delta^2$ denotes the pointwise union of δ^1 and δ^2. Ideally, we would like the individual security properties of p_1 and p_2 to carry over to $p_1 \cup p_2$.

First suppose that p_1 and p_2 are both secret-secure; we would like to show that this implies $p_1 \cup p_2$ is secret-secure. This is not possible in general because one protocol may interfere with the other. For example, consider the two (nonsensical) protocols in Fig. 4. In the first protocol, agent A shares its secrets with B by encrypting them with a shared key and sending them to B. The second protocol is similar, but here agent B shares secrets with agent A; the first message of the protocol is sent by A to indicate that it is "ready" to accept B's secrets. Both protocols are secret-secure since they do preserve secrets.[7] However, the union of these two protocols is not secret-secure. This is because the first step of the second protocol may send a message of the form $\{M\}_K$ such that M is some arbitrary message, and the second step of the first protocol may use this message to

7. Although they are not secure since nonces are not used to protect secrets, and as a result the protocols are not time-secure.

deduce that M is a shared $A - B$ secret, which is obviously not true in general. The problem is that messages of the form $\{(A, M)\}_K$ have incompatible uses in the two protocols. In the first protocol $\{(A, M)\}_K$ indicates that A believes M is a shared secret between A and B. In the second, $\{(A, M)\}_K$ indicates nothing about the secrecy of M.

$$p_1: \begin{cases} A: \ \mathit{shared}(\{A,B\},K),\ \mathit{shared}(\{A,B\},M) \to \mathit{send}(\{(A,M)\}_K) \\ B: \ \mathit{shared}(\{A,B\},K),\ \mathit{receive}(\{(A,M)\}_K) \to \mathit{shared}(\{A,B\},M) \end{cases}$$

$$p_2: \begin{cases} A: \ \mathit{shared}(\{A,B\},K) \to \mathit{send}(\{(A,M)\}_K) \\ B: \ \mathit{receive}(\{(A,M)\}_K),\ \mathit{shared}(\{A,B\},K),\ \mathit{shared}(\{A,B\},M') \\ \qquad \to \mathit{send}(\{(B,M')\}_K) \\ A: \ \mathit{shared}(\{A,B\},K),\ \mathit{receive}(\{(B,M)\}_K) \\ \qquad \to \mathit{shared}(\{A,B\},M) \end{cases}$$

Fig. 4. Two secret-secure protocols whose composition is not secret-secure.

Now, at an intuitive level, if protocols (P_1, δ^1) and (P_2, δ^2) do not "interfere" in this sense, then we may expect composition to preserve security properties. First, define for a protocol p, that *sends*(p) is the set of all messages sent in all secure models of p. More specifically, *sends*(p) is the set of all events of the form *send*(M) that appear in T, in some secure model $(T, \mathit{beliefs})$ of p. A form of noninterference can now be defined as follows:

DEFINITION 15. *Where $p_1 = (P_1, \delta^1)$ and $p_2 = (P_2, \delta^2)$ are protocols, p_1 is message independent with respect to p_2 if, for all sets* STATE *and $A \in P_1$, $\delta^1_A(\text{STATE}) = \delta^1_A(\text{STATE} - \mathit{sends}(p_2))$.*

Unfortunately this definition is not sufficient to prove preservation of secret-security. The failure is due to technical reasons arising from the unbounded nature of models. However, by restricting attention to directed models we can obtain a composition theorem. The following result shall additionally use two conditions: if $(T, \mathit{beliefs})$ is a secret-secure model of $p_1 \cup p_2$ then

1) all beliefs held by principals in the model have the form *shared*(S, M) where M is a basic message, and
2) for all secure directed models $(T, \mathit{beliefs})$ of $p_1 \cup p_2$, there exists a secure directed model $(T', \mathit{beliefs}')$ of $p_1 \cup p_2$ such that:
   - adversaries do not send any messages in T', and
   - for each event e in T_A, there exists an event e' in T'_A such that $\mathit{known}(e) \subseteq \mathit{known}(e')$, $\mathit{beliefs}(e) \subseteq \mathit{beliefs}'(e')$, and all beliefs in $\delta(\mathit{state}(e))$ also appear in $\delta(\mathit{state}'(e'))$.

The first condition says that, in all secure models, there are no beliefs involving compound messages or non-principals. This condition simplifies part of the proof; although it is likely that this condition could be weakened (or perhaps eliminated), it is already satisfied by typical protocols. The second states that, unless security is violated, the messages from adversaries do not have a significant effect on the behavior of a protocol. Specifically, it specifies that for any secure model, there is a secure model where adversaries do not send messages such that this secure model is "equivalent" to the original model in the following sense: for each event e in the original model, there is an event e' in the new model with essentially the same behavior. These conditions involve reasoning about secure models only, and are typically much easier to verify than checking the security of the combined protocol $p_1 \cup p_2$.

THEOREM 2. *Let p_1 and p_2 be protocols that are mutually message independent and satisfy conditions 1) and 2). If p_1 and p_2 are directed secret-secure then their composition $p_1 \cup p_2$ is also directed secret-secure.*

PROOF. Suppose that $p_1 \cup p_2$ is not directed secret-secure. Then there exists a directed model of $p_1 \cup p_2$, call it m, that is initially secure, but not universally secure. The proof now proceeds by using the model m to construct a model showing that either p_1 or p_2 is not secret-secure. This construction proceeds in three steps.

The first step uses condition 2) to show that there exists a model m' of $p_1 \cup p_2$ such that

- m' is initially secure but not universally secure;
- there are only a finite number of events in m';
- adversaries do not send any messages in m'.

The second step uses the model m' to construct a model m'' that satisfies the same conditions as m' and in addition does not contain any message generation or expiry events. The purpose of this step is to simplify the cases that must be considered in the following constructions.

The final and most intricate step uses the model m'' to construct either 1) a model of p_1 that is initially secure but not universally secure or 2) a model of p_2 that is initially secure but not universally secure. This part proceeds by incrementally constructing a model m_1 of protocol p_1 and m_2 of protocol p_2. In essence, each event from m'' is considered in turn (using the total ordering on events in m'' that is required to exist by the serializability assumption), and either adding it to m_1 (if the event corresponds to protocol p_1) or to m_2 (if the event corresponds to protocol p_2). Additional constructions are required at each step to ensure that m_1 and m_2 are maintained as legal, initially secure models of p_1 and p_2. Importantly, when all events from m'' have been considered, it is the case that either m_1 or m_2 is not universally secure. It follows that either p_1 or p_2 is not secret-secure. The key assumption used in this final step is the message independence assumption. We remark that condition 1) is used throughout the proof. □

To conclude this section, consider the case where p_1 and p_2 are time-secure. The compositionality results attainable in this case appear to be much weaker than for secret-security. One of the key problems is illustrated by the two protocols in Fig. 5. In the first protocol, agent A shares its secrets with B by encrypting them with a shared key and sending them to B. The second protocol is the converse of the first. The combined protocol $p_1 \cup p_2$ is not time-secure because of the following type of scenario: agent A gives secret M to B; the secret M then expires at A; B gives M back to A; the secret M then expires at B; A gives secret M to B, and so on. Any compositionality results involving time-security must clearly involve conditions to prevent the types of circularity that are illustrated in the example.

$$p_1: \begin{cases} A: \; shared(\{A,B\},K),\; shared(\{A,B\},M) \rightarrow send(\{(A,M)\}_K) \\ B: \; shared(\{A,B\},K),\; receive(\{(A,M)\}_K) \rightarrow shared(\{A,B\},M) \end{cases}$$

$$p_2: \begin{cases} B: \; shared(\{A,B\},K),\; shared(\{A,B\},M) \rightarrow send(\{(B,M)\}_K) \\ A: \; shared(\{A,B\},K),\; receive(\{(B,M)\}_K) \rightarrow shared(\{A,B\},M) \end{cases}$$

Fig. 5. Two time-secure protocols whose composition is not time-secure.

10 Towards Model Checking

Model theoretic frameworks open a new line of attack for proving security properties: model checking [7], [8], [9]. In this paper, security is defined in terms of properties that hold over a class of models. In principle, one could check the security of a protocol by constructing each model in the class, and checking to see that the appropriate property holds. Clearly, this procedure is not effective because there are an infinite number of models that must be checked, and these models themselves can be infinite. However, for certain syntactic classes of protocols, the problem of checking protocol security may be reduced to looking at a subclass of models whose behavior can be finitely represented using set constraints [15]. The key property of this subclass of models is that if the protocol is secure in this subclass, then it is secure in all models.

Appendix A
Variations of the Security Definitions

We provide a justification for the abstract nature of our model definition by showing that different and more concrete notions of model do not lead to different notions of security.

A.1 Ordered Lifetimes

We begin by considering a variation of our model definition where the order of belief expiration is that same as belief generation. To formalize this, let $(T, beliefs)$ be a model of a protocol P. An *event segment* for an agent A is a subset of the events in T_A such that if e_1 and e_2 are in the subset, then for any $e \in T_A$ such that $e_1 \leq_A e \leq_A e_2$, it must also be the case that e is in the subset. In other words an event segment is a contiguous subsequence of the events in T_A. Note that segments are always finite because of the boundedness condition (Definition 2). An event segment L is a *A-lifetime* (or just *lifetime*) for a belief b if b is held by A at each event in L, and L is a maximal segment with this property (that is, b is not held by A at the events immediately preceding L and immediately following L). In general, a belief may have several A-lifetimes. A model $(T, beliefs)$ *respects belief lifetimes* if, for each protocol principal A, whenever L_1 and L_2 are A-lifetimes then $L_1 \subseteq L_2$ implies $L_1 = L_2$. Intuitively, this says that the expiry of beliefs respects the order of belief establishment: if a belief b is established before b' then b will expire before b'.

Now, define alternative notions of secret-secure and time-secure by restricting the standard definitions to models that respect belief lifetimes. Such definitions are equivalent to the original definitions. This is because any model that is not secure can be modified by extending the lifetimes of the various nonces and beliefs (and possibly adding some extra dummy events) in such a way that 1) the modified model is not secure (it essentially has the same structure as the original model) and 2) the lifetime condition is satisfied.

A.2 Global Clocks

We next consider a more concrete definition of time using global clocks and bounds on the expiry of nonces and beliefs. Suppose that each agent is equipped with a clock and that these clocks are synchronized. Each event will be timestamped (with a real number) according to the time that it occurred. This timestamping respects message sending/receiving: if a message M is received by some agent at some time t, then M must have been sent by some agent at time t', for some $t' < t$. Further suppose that each agent holds a belief for no more than some fixed time ϵ before it expires. We call these *global clock models.* (It is also possible to consider models where there is some lower bound on lifetimes, or alternatively where there is a fixed lifetime for all secrets and nonces. In the latter case we could need to allow simultaneous events.)

Using global clock models, we can give definitions of secret-security and time-security analogous to those given in the main part of this paper. These resulting definitions of protocol security are in fact equivalent to the original definitions. This is because given any global clock model that is not secure, we can construct a standard model that is not secure, and conversely, given an standard model that is not secure, we can construct a global clock model that is not secure. The first part is easy, because given any global clock model, we can just ignore the timing information and obtain a standard model with the same essential structure.

Showing the converse is more difficult. Suppose that we have a standard model that is not secure. By definition, there must be some total ordering $<$ of all of the events in the standard model such that message receives are preceded by message sends (see Definition 3). This ordering forms the basis of our construction of a global clock model. Let T denote the trace of all events in the model, ordered by $<$.

What we need to do is associated a time with each event in the trace T, and the problem is how to do this in such a way that the expiry timing constraints are satisfied. In particular, what happens when the generation and expiry of beliefs is nested: that is, when belief b_1 is established at time t_1 and expires at time t'_1, belief b_2 is established at time t_2 and expires at time t'_2, and $t_1 < t_2 < t'_2 < t'_1$.

We begin with some definitions. We previously defined the notion of a segments for an agent A as certain subsets of T_A. We now generalize this definition and define T-segments to be subsets of T such that if two events are in the segment, then so are all events in between.

We now show that each event in T can be associated with a T-segment of events such that the assignment of times to the events in the T-segment can be done essentially independently of the assignments to events outside the T-

segment. Let $e \in T$ and consider all of the beliefs b held at e. Each belief can be identified with a lifetime L_b where $e \in L_b$. For each such lifetime, there is a minimal T-segment that contains L_b. Pick a T-segment that is maximal among all such T-segments, and denote it by $maxseg_e$.

Now, consider the strategy for assigning times to a T-segment of events $e_1, e_2, \ldots, e_n$ (where $e_1 < e_2 < \cdots < e_n$). We assume that either 1) some (possibly empty) initial subsequence of the T-segment already have times, and the remaining elements do not, or 2) some (possibly empty) final subsequence of the T-segment already have times, and the remaining elements do not. In the first case, suppose that $e_1, \ldots, e_i$ already have times $t_1, \cdots, t_i$, for some i, $1 \leq i \leq n$, and let $d = (\epsilon - (t_i - t_1))$ and assign times $t_i + (j \times d)$ to e_j, $i < j \leq n$ (so that the times assigned to $e_i, \ldots, e_n$ evenly divide up the interval $[t_i, t_1 + \epsilon]$). In the second case, suppose that $e_i, \ldots, e_n$ already have times $t_i, \cdots, t_n$, for some i, $1 \leq i \leq n$, and let $d = (\epsilon - (t_n - t_i))$ and assign times $t_i - (j \times d)$ to e_j, $1 \leq j \leq i$.

To complete the construction of a global clock model, we apply this strategy as follows. First, pick some event e from T and apply the strategy to $maxseg_e$. Then pick the event immediately following e (if it exists) and apply the strategy. Next, pick the event immediately preceding e (if it exists) and apply the strategy. Then repeat the process, alternately picking events succeeding and preceding e. This process is guaranteed to eventually reach all events in T (this follows from part 3) of Definition 3). Moreover, the use of $maxseg_e$ ensures that the time assignment strategy can be successfully applied (importantly, we never consider a T-segment that is completely subsumed by another T-segment).

We conclude this section with an observation about global clock models. In particular we address the question of whether, in the context of global clock models, the definition of secure protocol yields any explicit timing guarantees. The models used in the main part of this paper ensure only that a belief will eventually expire. Hence, if an agent A shares a secret with some other agents using a time-secure protocol, then A is guaranteed that the other agent's beliefs about the secret will eventually expire, but no bound is placed on when this will happen. However, using global clock models, we might expect that A has tighter guarantees about the lifetime of the secret. This is not the case in general because the secret may be passed from one agent to another. Each such pass gives rise to a (worst-case) ϵ delay in the expiry of the secret. We conjecture that for a given time-secure protocol there is some fixed N such that shared secrets are guaranteed to expire within time $N \times \epsilon$. We believe that for many protocols used in practice we can develop syntactic criteria to determine this index N.

ACKNOWLEDGMENTS

Nevin Heintze was supported in part by IBM through a graduate fellowship, and in part by the Defense Advanced Research Projects Agency, CSTO, under the title "The Fox Project: Advanced Development of Systems Software," ARPA Order No. 8313, issued by ESD/AVS under Contract No. F19628-91-C-0168. J. D. Tygar was supported in part by ARPA Contracts F33615-90-C-1465 and F33615-93-1-1330, National Science Foundation Presidential Young Investigator Grant CCR-8858087, matching funds from Motorola and TRW, a contract from the United States Postal Service, and by an equipment grant from IBM.

REFERENCES

[1] M. Abadi, "The power of temporal proofs," *Proc. Second IEEE Symp. Logic in Computer Science*, pp. 123–130, June 1987.

[2] M. Abadi and M. Tuttle, "A semantics for a logic of authentication," *Proc. of ACM Symp. Principles of Distributed Computing*, Aug. 1991.

[3] J. van Benthem, "Time, logic and computation," *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency.* J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, eds., Springer-Verlag, pp. 1-49, June 1988.

[4] P. Bieber, "A logic of communication in hostile environment," *Proc. IEEE Computer Security Foundations Workshop III*, pp. 14-22, Los Alamitos, Calif., June 1990.

[5] R. Bird, I. Gobal, A. Herzberg, P.A. Janson, S. Kutten, R. Molva, and C.M. Yung, "Systematic design of a family of attack-resistant authentication protocols," *J. Selected Areas in Comm.*, vol. 11, no. 5, pp. 679-693, June 1993.

[6] M. Blum and S. Goldwasser, "An efficient probabilistic public-key encryption scheme which hides all partial information," *Advances in Cryptology: Proc. of CRYPTO'84*, Springer-Verlag LNCS no. 196, 1984.

[7] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. "Symbolic model checking: 10^{20} states and beyond," *Information and Computation*, vol. 98, no. 2, pp. 142–170, June 1992.

[8] E.M. Clarke and E.A. Emerson "Synthesis of synchronization skeletons for branching time temporal logic," *Logic of Programs: Workshop, Lecture notes in Computer Science.* vol. 131. Yorktown Heights, N. Y.: Springer-Verlag, May 1981.

[9] E.M. Clarke, E.A. Emerson, and A.P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Trans. Programming Languages and Systems*, vol. 8, no. 2, pp. 244–263, 1986.

[10] J. Burgess, "Basic tense logic," *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, E. Gabbay and F. Guenthner, eds., pp. 89-133. Kluwer Academic Publishers, 1986.

[11] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Trans. Computer Systems*, vol. 8, no. 1, pp. 18-36, Feb. 1990. Also see Research Report no. 39, DEC SRC, 48 pp., 1989.

[12] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, Mar. 1983. Also appears in the 22nd FOCS, 1981.

[13] S. Even and O. Goldreich, "On the security of multi-party ping-pong protocols," *Proc. 24th IEEE Symp. Foundation of Computer Science*, pp. 34-39, Oct. 1983.

[14] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all private information," *Proc. 14th ACM Symp. Theory of Computing*, 1982.

[15] N. Heintze and J. Jaffar, "A decision procedure for a class of Herbrand set constraints," *Proc. Fifth IEEE Symp. Logic in Computer Science*, pp. 42-51, June 1990.

[16] T. Kasami, S. Yamamura, and K. Mori, "A key management scheme for end-to-end encryption and a formal verification of its security," *Systems, Computers, Control*, vol. 13, pp. 59-69, 1982.

[17] L. Lamport, "Time, clocks and the ordering of events in a distributed system," *Comm. ACM*, vol. 21, no. 7, pp. 558-565, July 1978.

[18] H.R. Lewis and C. Papadimitriou, *Elements of the Theory of Computation.* Prentice-Hall, 1981.

[19] C.A. Meadows, "Applying formal methods to the analysis of key management protocol," *J. Computer Security*, vol. 1, pp. 5-53, 1992.

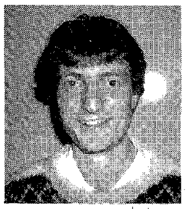
[20] M.J. Merritt, "Cryptographic protocols," PhD thesis, Georgia Instit. of Technology, Feb. 1983.

[21] D. Otway and O. Rees, "Efficient and timely mutual authentication," *Operating Systems Review*, vol. 21, no. 1, pp. 8-10, Jan. 1987.

[22] P. Syverson and C. Meadows, "A logical language for specifying cryptographic protocol requirements," *Proc. 1993 IEEE Symp. Research in Security and Privacy*, May 1993.

[23] P. Syverson, "The use of logic in the analysis of cryptographic protocols," *Proc. 1991 IEEE Symp. Research in Security and Privacy*, May 1991.

[24] P. Syverson "Adding time to a logic of authentication," *Proc. First ACM Conf. Computer and Comm. Security,* Fairfax, Va., Nov. 1993.



Nevin Heintze received the BSc and Msc degrees in mathematics and computer science from Monash University, Melbourne Australia, and the PhD degree in computer science from Carnegie Mellon University, Pittsburgh, Pennsylvania. He is now a member of the technical staff at AT&T Bell Laboratories, Murray Hill, New Jersey. His interests include security, programming languages, program analysis, and types. He developed the Set-Based Analysis system for ML.

J.D. Tygar (M'82) is an associate professor of computer science at Carnegie Mellon University. He received an AB degree from the University of California at Berkeley and his PhD degree from Harvard University. His recent research focuses on computer security and electronic commerce. His current projects include the Dyad secure coprocessor system, the NetBill electronic commerce systems (directed jointly with Marvin Sirbu), and cryptographic postal indicia standards. He was named a Presidential Young Investigator. He consults widely for industry and government.