A Model for Secure Protocols and Their Compositions

(extended abstract)

Nevin Heintze and J. D. Tygar

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213-3890.

Email: nch@cs.cmu.edu, tygar@cs.cmu.edu

Abstract

We give a formal model of protocol security. 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.

I. 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 family of tools for reasoning about protocol security. We adopt a model-based approach for defining protocol security properties. This allows us to describe security properties in much greater detail and precision than previous frameworks for reasoning about protocol security.

Some of the most advanced work in this area is represented by a paper by Burrows, Abadi, and Needham [8], which presents a proof-based (or rule-based) approach to reasoning about security properties of protocols. (Also see [2].) 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.

The 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 sub-protocol, or A and B may be concatenated, or A and B may run simultaneously as co-protocols. 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 not 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 that are believed to be secret are never revealed. By time-security, we mean that stale messages can not 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 issues relating to time. Our paper considers these properties in a wide framework which makes them applicable to any reasonable notion of time.

Related Work

Time has received increasing attention in the literature on protocol security analysis. One of the first papers to consider time was [8], where one could reason about the freshness of nonces and messages. However the notion of time here 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 detail by [5].

This paper considers a purely model-based view. The idea of using models to reason about security has been explored in a number of papers. Early work by Dolev and Yao [9] 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 [15], [18], [17] considered more general models that address a much larger class of protocols and a wider variety of security properties. We note that the model described in [18] is based on a trace of the 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.

Despite all of this work on models, only a few papers have considered security models that deal directly with the issue of time. [2] use model theoretic constructions to clarify and explain the Logic of Authentication. The structure of the model is closely tied to the logic; it is intended to illustrate the logic rather than provide an independent description of security properties. Based on this model, [22] 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 message sends and receives for 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. [20] use a logic language equipped with temporal operators for specifying protocol requirements to be checked by the NRL Protocol Analyzer [17]. Most of these papers address only limited aspects of time. Many assume a global synchronized notion of time, do not consider interactions between different instances of protocols, and do not address issues relating to the expiry or nonces and beliefs.

To summarize, the following aspects of our paper are new:

- 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).
- An explicit account of agent beliefs and nonces, including generation and expiry.
- A composition theorem on secure protocols.

II. The Case for Models

The literature on reasoning about protocol security can broadly be classified into two camps: works that are proof-based (that is, they proceed by constructing "proofs" in a logic designed for reasoning about security) and model-based (that is, they reason about some algebraic — and typically idealized — model of how agents can interact).

Proof-based approaches 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 $A \stackrel{K_{AB}}{\longleftrightarrow} B$ to denote the property that K_{AB} is a secure key for communication between A and B, and the compound formula $A \models A \stackrel{K_{AB}}{\longleftrightarrow} B$ 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

$$\frac{A \models A \overset{K_{AB}}{\longleftrightarrow} B, \quad A \triangleleft \{X\}_{K_{AB}}}{A \models B \mid\!\sim X}$$

where $A \triangleleft \{X\}_{K_{AB}}$ denotes that agent A sees the message X encrypted with key K_{AB}, and $A \models B \mid\!\sim X$ 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.

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 Burrow, Needham and Abadi'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? Proof-based 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 very 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 are establishing 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 plaintext.
- 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 focus on how agents interact with each other: what messages they can see, what messages they can decrypt and encrypt, what messages they can send, etc. Various security properties can then be defined in terms of these interactions. For example, we can ask what messages can be discovered by an adversary.

The specification of agent interactions is therefore central to the definition of a security model. It should include all physically possible interactions, and at the same time exclude "impossible behaviors" (such as when an adversary "guesses" secrets). A comprehensive security model should also reason about time, including the evolution of agent beliefs (that is, the establishment and expiry of beliefs) and the behavior of nonces.

Such a specification can be developed by viewing agents as non-deterministic machines that communicate over a public channel. These machines have some internal state (such as a set of known keys and messages, and some beliefs about them), and various constraints limiting their behavior — some are based on assumptions relating to physical considerations, and others are based on the protocol at hand. Example constraints are: (a) a message must be sent before it is received (this assumption is based on physical considerations); (b) secrets can not be guessed (this is primarily an assumption about the integrity of the underlying encryption method); and (c) trusted agents send messages in accordance with the protocol. Note that the process of defining a model requires that we explicitly identify the assumptions that are to be included in the model; there is, of course, considerable choice about what assumptions we include and exclude from a particular model.

While model-based approaches enjoy a number of desirable properties they do not provide direct reasoning principles. The problem is that 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 justifying proof-based-systems, but they can also be used to develop model-checking methods for protocol verification.

The above descriptions of model-based and proof-based approaches emphasize their differences. However, the distinctions are often not so clear cut in the literature. This is partly because there are a variety of views of what constitutes a model-based approach and a proof-based approach. Moreover, a number of papers use a combination of both approaches. For example Abadi and Tuttle [2] use a model-based construction to explain various aspects of the Logic of

Authentication. However this model does not attempt to give an independent definition of the notions of security and belief contained in the logic. Rather, it is heavily based on the structure and concepts of the logic.

In contrast, our work seeks to establish an independent notion of security by developing a self-contained model from first principles that explicitly identifies assumptions.

III. 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. In essence 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], [11].

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 will have to 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 will be insecure regardless of the encryption scheme used. In practice, this has lead to the discovery of a number of problems in published protocols. Another view of the perfect encryption assumption is that it provides a way to decompose the problem of analyzing a protocol into two sub-problems: (a) an analysis of the protocol that focuses on the intrinsic security properties of the protocol by ignoring details of the encryption scheme used, and (b) 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 non-decomposable[1] message of the model[2]. Notationally, we shall use K to denote elements of $\mathcal{B}$ that are used as keys, and N to denote

[1] That is, messages that are not encryptions or compositions of other messages.

[2] We will assume that each agent A has a publically known name (also denoted A) which is a basic message.

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 de-compose these messages into their constituents and also built up new messages by encryption and composition. Specifically, define $S^\star$, the set of messages *deducible* from S, to be either (i) any element of S, (ii)[3] M such that $\{M\}_K$ and K are in $S^\star$, (iii) M_i such that $(M_1, \ldots, M_n)$ is in $S^\star$, (iv) $\{M\}_K$ such that M and K are in $S^\star$, and (v) $(M_1, \ldots, M_k)$ such that each M_i is in $S^\star$.

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:

$$decompose_i(M_1, \ldots, M_n) = M_i, \quad 1 \leq i \leq n$$

$$decrypt\,(K, \{M\}_K) = M.$$

IV. 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). Each agent shall be viewed 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 (message sends and receives) 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 if necessary to indicated different occurrences of the same event. This is needed, for example, if an agent resends a message.

[3]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 $S^\star$.

Intuitively, a trace can now be defined as 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 (for example, a real number) with each event. Unfortunately, such an approach assumes the existence of a global clock. In an insecure 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 non-standard notions of time such as Lamport clocks [16]. This is based on a very minimal assumption about time: each agent has a *local* notion of "before" and "after" that defines a total order on each agent's events. Then:

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

We use T_A to denote the set corresponding to the agent A, and T to denote 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 pre-order. However, it can be shown that 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 [3], [7], [1].)

We remark that an agent typically sends a message to some designated recipient (such information may be implicit in a protocol, or explicit in the plaintext or encrypted parts of messages). However, we shall treat the communication medium as insecure, and so we shall not guarantee that a message is received only by its intended receiver. Rather, we shall 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* <u>bounded</u> *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 [18]). Moreover, we require that this interleaving must be "fair".

Definition 3 (Serializable Traces) *A trace T is* <u>serializable</u> *if there is a total order $<$ on T such that*

(a) the ordering $<$ conservatively extends the orderings $<_A$, $A \in \mathcal{A}$, in the sense that if $e <_A e'$ then $e < e'$,

(b) each event receive(M) in T is preceded by an event send(M), and

(c) for all pairs of events e_1 and e_2, the set $\{e : e_1 < e < e_2\}$ is finite. □

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

Definition 4 (Message Generation) *A trace T* <u>respects message generation</u> *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.

V. 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, the following is a description of variant of the Otway-Rees protocol [19].

$A \rightarrow B$: $A, B, \{N_A, B\}_{K_A}$
$B \rightarrow S$: $\{N_A, B\}_{K_A}, \{N_B, A\}_{K_B}$
$S \rightarrow B$: $\{N_A, B, K_{AB}\}_{K_A}, \{N_B, A, K_{AB}\}_{K_B}$
$B \rightarrow A$: $\{N_A, B, K_{AB}\}_{K_A}$

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 repond with 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 $N_A, N_B, K_A, K_B, \ldots$ 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 (i) what the agent currently believes (about keys and nonces) and (ii) an event (such as the receipt of a message).

We first formalize the notion of beliefs used in our model. We consider two kinds of beliefs: beliefs about secrecy and beliefs about freshness. Many other kinds of beliefs could be included, however these two are central to the treatment of time, which is the main focus of our work. We believe that our model can be extended to incorporate additional beliefs, but we leave an investigation of this to future work. A *belief* is of the form $\mathit{shared}(S, M)$ where S is a set of agents and M is a message, or of the form $\mathit{fresh}(M)$ where M is a basic message. The first belief indicates that the agent considers M to be a message that is only known by the agents in S. The second belief indicates that the agent considers M to be fresh.

Next, we define the set of messages that an agent knows about: it is from this set that an agent can send messages and construct new beliefs. Let STATE be a set of beliefs and events (intuitively, this represents an agent's current set of beliefs, and the events that it has experienced thus far). Define $\mathit{known}(\text{STATE})$, the messages that can be constructed from the information present in

STATE, to be the following set:

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

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 that can be derived from the above messages.

Now, given an agent A, a set of beliefs and a set of events, a protocol specifies a set of possible actions that an agent can take. Such actions include sending messages and adding new beliefs to the agent'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* $(\delta_A,\ A \in P)$. *Each function* δ_A *maps any set consisting of receive events, send events*[4]*, 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* $\delta_A(\text{STATE})$ *is a set consisting of the following two kinds of elements:*

(a) beliefs of the form $\mathit{shared}(S, M)$ *where* $M \in \mathit{known}(\text{STATE})$*, or*

(b) events of the form $\mathit{send}(M)$ *where* $M \in \mathit{known}(\text{STATE})$*.* □

Elements of the form (a) in $\delta_A(\text{STATE})$ indicate new beliefs that A should add to its set of current beliefs. Elements of the form (b) indicate messages that A should send. We shall assume that each function δ_A is monotonic. Note that we have not included generate and expiry messages in the notion of "state" used in the above definition. The meaning of these events shall be defined explicitly in the model, and shall be independent of the protocol at hand.

A trace is *faithful* to a protocol if each principal behaves in accordance with the protocol. Now, 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 $\mathit{expiry}(M)$ 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 have thus far strived to avoid introducing an explicit notion of time. To maintain this level of abstraction,

[4] The functions δ_A are usually independent of the send elements in STATE, since protocol actions rarely depend on previous message sends.

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.*

Before presenting the definition of model, 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* model *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*

(a1) If e is $send(M)$ then $e \in \delta_A(state_A(e))$.

(a2) If e is $expire(M)$ then

- $beliefs(succ(e)) = beliefs(e) - \text{BELIEFS}_M$.

(a3) 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 $S \subseteq \mathcal{A}$.*

(a4) If e is $send(M)$ or $receive(M)$ then

- $beliefs(e) \subseteq beliefs(succ(e))$, *and*
- $beliefs(succ(e)) - beliefs(e) \subseteq \delta_A(state_A(e) \cup \{e\})$.

and if $A \notin P$ then:

(b1) $beliefs(succ(e)) \subseteq known(state_A(e) \cup \{e\})$.

(b2) If e is $send(M)$ then $M \in known(state_A(e))$. □

In items (a2), (a3), (a4) and (b1), if $succ(e)$ does not exist, then the condition is vacuously true. Items (a1-a4) constrain the behavior of protocol principles; items (b1-b2) constrain the behavior of adversaries. We discuss each collection in turn.

Item (a1) states that the messages sent by a protocol principal must be in accordance with the protocol. In combination with part (b) of the definition of protocol, it also implies that principles can only send messages that they know about. Items (a2-a4) 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 be in accordance with the protocol. Note that in (a4), 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 $\delta_A(state_A(e) \cup \{e\})$. Note that, 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 (b1-b2) address non-principles. Item (b1) states that the beliefs of an adversary can only grow by either message generation or by the receipt of messages. Item (b2) 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 (b1) and (b2) in fact hold for all agents: for protocol principles, the conditions expressed by (b1) and (b2) are consequences of the definition of protocol and items (a1-a4).

We now address the meaning of beliefs. An 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. Hence, ignoring the time component of their meaning, beliefs of the form *fresh*(M) shall simply be considered to be *true*. Second, consider beliefs of the form *shared*(S, M). The meaning of these beliefs is dependent on what 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* <u>valid</u> *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 this 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 non-principals. 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 basically cross-sections of the model.

Definition 9 (Snapshots) *In the context of a model* $(T, \mathit{beliefs})$, *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 $\mathit{beliefs}(s(A))$ 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 $b \in \mathit{beliefs}(e)$, and for each event $e' <_A e$, it is also the case that $b \in \mathit{beliefs}(e')$. 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 include the starting keys that agents use for talking to servers. Now, the notion of a model being started in an initially secure state shall be realized by placing properties on the set of initial beliefs. Clearly the 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 $\mathit{shared}(\{A_1, B_1\}, M)$ and the agent A_2 has the initial belief $\mathit{shared}(\{A_2, B_2\}, M)$, 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 to specify. We therefore require that initial beliefs must not contain other messages; that is, initial beliefs must only involve basic messages.[5]

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

[5]This restriction can be weakened somewhat; however since initial beliefs are typically used only to establish starting keys, this does not appear to be necessary.

Note that 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. This is required because there may be messages in "transit" that are not represented by a single snapshot. Finally, we can define protocol security in terms of the preservation of security:

Definition 11 (Secret-Security) *A model $(T, \mathit{beliefs})$ is* <u>secret-secure</u> *if, for all principals A of P and for all $e \in T_A$, each element of $\mathit{beliefs}(e)$ is valid at each event in T. A protocol (P, δ) is* <u>secret-secure</u> *if all initially-secure models for (P, δ) are secret-secure.* □

This definition provides a very 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 to reason about the freshness of nonces. In doing so, we obtain a definition of protocol security that supports reasoning about security compromises that are based on the replay of stale messages. We conclude this section with a discussion of 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 <u>directed secret-secure</u> *if all initially-secure directed models for the protocol are universally 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. Recall that 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

[6]However, the two definitions coincide for most "reasonable" protocols.

agent can send, because the only beliefs that are 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 secret-secure protocol. By definition, this implies that $shared(S, M)$ 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, agents A and B will only participate if they believe that their initial keys for talking to the server 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 principles.

VI. 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 insure that certain information is fresh. Furthermore, it is expected that propositions such as "this key is a secure key" do not hold forever, but are at some time considered to expire when the key becomes stale.

First, 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 considered fresh until time $t + \delta$. Of course, δ may vary from nonce to nonce. Instead of committing to a specific mechanism, we employ a very simple and abstract characterization: each nonce eventually expires. This is formalized by requiring that if an agent holds a belief of the form $\mathit{fresh}(M)$ at some event, then there exists a later event of the form $\mathit{expire}(M)$. Since the only mechanism for an agent to add $\mathit{fresh}(M)$ to its set of beliefs is via an event of the form $\mathit{generate}(M)$, 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, an important aspect of their behavior is that 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 re-established at yet another later time. An appropriate expiry condition for beliefs is: if an agent holds a belief of the form $\mathit{shared}(S, M)$ at some event, then there exists a later event of the form $\mathit{expire}(M)$. 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* timed model *if for each principal A of P and event $e \in T_A$, if $\mathit{beliefs}(e)$ contains a belief of the form $\mathit{shared}(S, M)$ or $\mathit{fresh}(M)$ then there exists an event $e' > e$ such that e is $\mathit{expire}(M)$. A timed model is* time-secure *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* time-secure *if each* *initially secure timed model for P is time-secure.* □

We now discuss why this captures an important notion of security. An important failure mode of a protocol 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 not be considered insecure. 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 between 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 protocol insecurity.

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 secure if it is both secret-secure and time-secure.* □

We now provide some 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 then address the issue of whether the resulting definition is of general applicability, because the notion of security itself may be dependent 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.

We believe that our definition of security is essentially equivalent to any reasonable definition in the context of a specific notion of time. In other words, we claim that our definition captures

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

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

Fig. 1. Two secret-secure protocols whose composition is not secret-secure.

the essence of what it means for a protocol to be secure. In an earlier report [14], we provided evidence for this assertion by comparing three models with different notions of time: one was an early version of the model presented in this paper, and the other two differed 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 sub-classes 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:*

(a) Is P secret-secure?

(b) Is P time-secure?

Proof: We have defined protocols in a very general kind of transition system. As a result, it is easy to code Post's correspondence problem as a protocol such that a belief of the form $\mathit{shared}(\{A\}, M)$ 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 semi-secure secure if and only if the correspondence problem has a solution. This outline proves (a). 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 have been based on complex algebraic properties of much richer encryption systems. For example the proof of undecidability for *half-word ping-pong* protocols [10] relies critically on the structure of a more general encryption-decryption scheme that allows commutativity of some pairs of operators.

VII. 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 clearly not possible in general because one protocol may interfere with the other. For example, consider the two (rather nonsensical) protocols in Figure 1. 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 very 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. While neither protocol is really secure (in particular, nonces are not used to protect secrets, and as a result the protocols are not time-secure), they do preserve each other's secrets and can be shown to be secret-secure. 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 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.

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 $\mathit{sends}(p)$ is the set of all messages sent in all secure models of p. More specifically, $\mathit{sends}(p)$ is the set of all events of the form $\mathit{send}(M)$ that appear in T, in some secure model $(T, \mathit{beliefs})$ of p. A form of non-interference 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 that relate to the unbounded nature of models. However, by restricting attention to directed models we can obtain a compositionality theorem. The following result shall additionally use the following two conditions: if $(T, \mathit{beliefs})$ is a universally secure model of $p_1 \cup p_2$ then

(a) all beliefs held by principals in the model have the form $\mathit{shared}(S, M)$ where M is a basic

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

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

Fig. 2. Two time-secure protocols whose composition is not time-secure.

message, and

(b) 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:

(b1) adversaries do not send any messages in T', and

(b2) 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 event, there is an event e the new model with essentially the same behavior. These conditions involve reasoning only about secure models, and as such 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 (a) and (b). 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 (b) 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 step and most intricate step uses the model m'' to construct either (i) a model of p_1 that is initially secure but not universally secure or (ii) 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 (a) is used throughout the proof. $\square$

To conclude this section, consider the case where p_1 and p_2 are timed-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 Figure 2. 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 just the converse of the first. The combined protocol $p_1 \cup p_2$ is not timed-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 above example. We leave further details to future work.

VIII. Future Work: Model Checking

The model theoretic framework we have described above opens a possible new line of attack for proving security properties. 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 hold. 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 [13]. 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. (For a general discussion of the model checking versus other methods, see [12].)

Acknowledgments

The first author 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. The second author was supported in part by ARPA contracts F33615-90-C-1465 and F33615-93-1-1330, NSF Presidential Young Investigator Grant CCR-8858087, matching funds from Motorola and TRW, a contract from the US Postal Service, and by an equipment grant from IBM.

References

[1] M. Abadi, "The Power of Temporal Proofs", *Proceedings* 2^{nd} *IEEE Symposium on Logic in Computer Science*, pp. 123–130, June 1987.

[2] M. Abadi and M. Tuttle, "A Semantics for a Logic of Authentication", *Proceedings of ACM Symposium on Principles of Distributed Computing*, August 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" *IEEE Computer Security Foundations Workshop III*, pp. 14-22, Los Alamitos, California, June 1990.

[5] R. Bird, I. Gobal, A. Herzberg, P. A. Janson, S. Kutten, R. Molva, M. Yung, "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *Journal on Selected Areas in Communications*, 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: Proceedings of CRYPTO84*, Springer-Verlag LNCS No. 196, 1984.

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

[8] M. Burrows, M. Abadi and R. Needham, "A Logic of Authentication", *ACM Transactions on Computer Systems*, Vol. 8, No. 1, pp. 18-36, February 1990. (Also see Research Report

No. 39, DEC SRC, 48 pages, 1989.)

[9] D. Dolev and A. C. Yao, "On the Security of Public Key Protocols" *IEEE Transactions on Information Theory*, Vol. IT-29, No. 2, March 1983. (Also appears in the 22^{nd} FOCS, 1981.)

[10] S. Even and O. Goldreich, "On the Security of Multi-Party Ping-Pong Protocols", *Proceeding 24^{rd} IEEE Symposium on the Foundation of Computer Science*, pp. 34-39, October, 1983.

[11] S. Goldwasser and S. Micali, "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Private Information", *Proceedings 14^{th} ACM Symposium on the Theory of Computing*, 1982.

[12] J. Y. Halpern and M. Y. Vardi, "Model Checking vs. Theorem Proving: A Manifesto", Systems." *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, February 1991.

[13] N. Heintze and J. Jaffar, "A Decision Procedure for a Class of Herbrand Set Constraints", *Proceedings 5^{th} IEEE Symposium on Logic in Computer Science*, pp. 42-51, June 1990.

[14] N. Heintze and J. D. Tygar, "Timed Models for Protocol Security", Carnegie Mellon University Technical Report CMU-CS-92-100, January 1992.

[15] 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.

[16] L. Lamport, "Time, Clocks and the Ordering of Events in a Distributed System", *Communications of the ACM*, Vol. 21, No. 7, pp. 558-565, July 1978.

[17] C. A. Meadows, "Applying Formal Methods to the Analysis of a Key Management Protocol", Journal of Computer Security, 1:5-53, 1992.

[18] M. J. Merritt, Cryptographic Protocols, Ph.D. thesis, Georgia Institute of Technology, February 1983.

[19] D. Otway and O. Rees, "Efficient and Timely Mutual Authentication", *Operating Systems Review*, Vol. 21, No. 1, pp. 8-10, January 1987.

[20] P. Syverson and C. Meadows, "A Logical Language for Specifying Cryptographic Protocol Requirements", *Proceeding 1993 IEEE Symposium on Research in Security and Privacy*, May 1993.

[21] P. Syverson, "The Use of Logic in the Analysis of Cryptographic Protocols", *Proceeding 1991 IEEE Symposium on Research in Security and Privacy*, May 1991.

[22] P. Syverson "Adding Time to a Logic of Authentication", *Proceeding* 1^{st} *ACM Conference on Computer and Communications Security*, Fairfax, Virginia, November 1993.