Timed Models for Protocol Security

Nevin Heintze and J. D. Tygar

January, 1992

CMU-CS-92-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was partially sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597. N. Heintze was supported by an IBM Graduate Fellowship. J. D. Tygar was supported by National Science Foundation Presidential Young Investigator Award CCR-8858087.

Keywords: key management protocols, computer security, formal models

Abstract

The notion of time is prerequisite for describing and verifying the security properties of key management protocols. Without it, properties relating to the expiration of keys and the freshness of messages and nonces cannot be formulated. Recently Burrows, Abadi and Needham proposed a formal system for protocol verification which includes an ability to reason about time. In essence their "Logic of Authentication" is a proof theory for reasoning about key management protocols.

One difficulty with such a logic lies in justifying the inferences that can be made. We approach this problem by developing an accompanying model theory for protocol security. Model theoretic techniques have been used before in the protocol verification literature, but our approach is different in two respects. First we consider a model theory which includes a notion of time. Second, the purpose of much of the previous model theoretic work was aimed at developing protocol verification tools and so assumptions about specific kinds of protocols and methods for breaking protocols were built into the model, often implicitly. In contrast, our account is more general and centers on a justification of the notion of model itself.

The main results of this work include

- a model theoretic definition of protocol security that is provably equivalent to a variety of alternative definitions;
- demonstration that some questions about protocol security properties are undecidable, and
- a schema for demonstrating the validity of many protocols by the use of model checking.

1 Introduction

This paper lays the groundwork for evaluating claims about secure protocols. It presents a formal model of agent interaction in protocols, and defines notions of security relative to that model. This work is of both theoretical importance and of practical importance, because it gives us a way of justifying inference rules that might be used in verifying a protocol's security.

Suppose that two agents wish to conduct a secure conversation; that is they wish to prevent an adversary from (1) understanding their conversation and (2) confusing the conversation by inserting or deleting messages (includes re-sending previous legitimate messages). If we have an encryption method, we can often reduce the problem to a question of authenticating the parties involved and establishing a secure *session* key that can be used for the conversation.

Getting this right is not a simple matter! It is one of the most difficult problems in the theory of secure protocols. As has been widely observed, many published authentication and key distribution protocols have critical flaws in them. Perhaps no other area of distributed protocols has been so plagued by faulty protocols.

To insure that our protocols are sound, we must develop methods for describing and verifying the properties of key management protocols. Recently, an important tool was introduced by Burrows, Abadi, and Needham[**?**], a "Logic of Authentication" which acts as a proof theory for reasoning about these protocols. This work has practical importance, and has revolutionized our techniques for judging secure protocols. However, their logic can not be formally justified without further development; in particular, there is a question about demonstrating the validity of their inference rules. In addition, the logic uses assumptions that are not formally stated, and is incomplete in the sense that there are valid protocol properties which cannot be proved in the logic.

It is clear, however, that the logic of Burrows, Abadi, and Needham is a valuable tool for debugging protocols. This paper explores the boundaries of such logics. After reviewing and critiquing the logic, we begin by asking the question "How do we know when we have the correct inference rules in such a logic?" To build a framework for justifying inference rules, we must first develop a formal model of protocol interaction. Other researchers have suggested using model theoretic techniques for protocol verification [**?**, **?**, **?**], but our work differs from these works in two respects. First, we integrate time into our model. Second, most previous model theoretic work was directly focused on building protocol verification tools, and thus assumptions about specific types of protocols and methods for breaking them were built into the model, often implicitly. Our account is much more general and does not restrict an adversary to some fixed list of possible attacks. Then, we define precise notions of "security" and "belief" in that model and discuss methods about reasoning in that model. We conclude by discussing possibilities and limitations of using a model checking approach for security properties that involve time.

The main results of this work include

- a model theoretic definition of protocol security that is provably equivalent to a variety of alternative definitions;
- demonstration that some questions about protocol security properties are undecidable, and
- a schema for demonstrating the validity of many protocols by the use of model checking.

Our focus in this work is on a model of security where private key based methods are used. Our work extends with only a few changes to the use of public key based methods. Further extensions of our work could encompass a wide class of protocols. For example, while we only consider issues of first-level belief among agents in our system, work by Fagin, Halpern, Vardi and others [?, ?] has demonstrated that issues of belief can be quite subtle. It would be interesting to apply our techniques to a fuller notion of belief. Also, like previous researchers, we assume an idealized notion of cryptography where no partial knowledge can be leaked out of a cryptosystem, random events are truly random, etc. Real cryptosystems often do not observe these properties and several seemingly sound cyptographic protocols fail because partial knowledge can be leaked in the system [?, ?, ?]. In light of recent work on the possible attacks on the DES cryptosystem [?, ?], an extension of our analysis may be very important in actual applications. Another important extension of this model would allow it to account for zero-knowledge authentication protocols [?, ?]. This work builds a structure which we believe can be extended to all these applications, and moreover would allow these protocols to be verified through model checking approaches.

2 Summary and Critique of "Logic of Authentication"

The "Logic of Authentication" framework [?] reasons at the level of the beliefs of agents. The framework consists of two components: (i) a language for expressing judgements about beliefs, and (ii) a collection of rules for reasoning about these judgements. The basic forms of judgements are $A \models X$ and $A \triangleleft X$, where A is an agent and X is some message. The intended meaning of the first is that A believes X or rather A is entitled to believe X, or, perhaps more appropriate to its use in the logic, A can prove X (that is, from A's assumptions and interactions with other agents, it can proved that X is a valid message or a valid judgement). The intended meaning of the second is that is that A sees the message X. In both of these forms of judgment, the message X can be either some plaintext message, or a formula of the logic[1]. In particular X can be another judgement. For example $A \models (B \models X)^2$ denotes the judgement that A believes that B believes X.

A number of other constructions are available to build up a variety of different kinds of judgements. For example $A \models B \mid\sim X$ denotes the judgement that A believes that B once said X, and $A \models A \overset{K_{AB}}{\longleftrightarrow} B$ denotes the judgement that A believes that K_{AB} is a secure key for conversation with B. Corresponding to these judgements are 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 $\{X\}_{K_{AB}}$ denotes the encryption of X with the key K_{AB}. This rule states that if A believes that K_{AB} is a secure key for conversations with B, then A believes that any message which is encoded by K_{AB} must have originally been sent by B[3]. Note that A has no reason to believe that the message X was sent recently. An intruder could have stored $\{X\}_{K_{AB}}$ and then replayed it.

One of the most important kinds of judgments involves whether a message is fresh. The judgement $A \models \#(X)$ denotes the judgement that A believes that X is fresh. Typically, the freshness of a message is guaranteed through the generation and judicious use of unique identifiers called nonces. For example the judgement $A \models \#(X)$ is appropriate if X is a message which contains a nonce recently created

[1] In [?], messages are simply identified with statements in the logic.

[2] In what follows, we omit parenthesis where there is no ambiguity.

[3] It is implicitly assumed that X was not sent by A.

by A. An important inference rule involving this judgement is[4]:

$$\frac{A \models B \mid\sim X, \quad A \models \#(X)}{A \models B \models X}$$

In other words, if A believes that B once said X and X is fresh then A believes that B believes X. Note that this reasoning identifies messages recently sent by an agent with that agent's beliefs. This does not mean that A necessarily believes everything that B says, but rather that A's view of B's beliefs are taken to be exactly the messages recently sent by B to A.

The above logic cannot be applied directly to a analyze a protocol. Instead the protocol must first be expressed in a form which is compatible with the structure of the logic. This form consists of two parts. The first part, called the "idealized" protocol, is the representation of the protocol itself. Essentially, it is obtained from the protocol by removing all plaintext components and also making explicit the assertions which are implicit in transmitted messages. The second part states the starting conditions of the protocol.

Consider the following simple protocol, where A and B are two agents who wish to establish a secure session key and S is a key server.

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

The purpose of the second message of this protocol is for S to convey to A that K_{AB} is a secure key for an A–B conversation. In the idealized protocol, this message is written as $\{A \overset{K_{AB}}{\longleftrightarrow} B\}_{K_{AS}}$, making the assertion explicit. The starting conditions for this protocol include that A believes that K_{AS} is a secure key for conversation with S, that is, $A \models A \overset{K_{AS}}{\longleftrightarrow} S$. This means that when A receives message 2 ($A \triangleleft \{A \overset{K_{AB}}{\longleftrightarrow} B\}_{K_{AS}}$) we have, on applying the first inference rule given previously, that $A \models S \mid\sim \{A \overset{K_{AB}}{\longleftrightarrow} B\}_{K_{AS}}$. In turns out that this is the strongest statement that A can prove regarding the key K_{AB}. This reflects the obvious weakness of this simple protocol: on receiving a protocol message, no agent can be sure that the message is recent, and hence there is no guarantee that the key contained in the message is currently a secure key.

The remainder of this section consists of a brief analysis of the Logic of Authentication framework. We ask two questions: "Are inferences in the logic valid?"

[4] A side condition on this rule is that X does not contain any encrypted messages.

and "Are all valid formulas of the logic provable?"

First, consider the question of the validity of inferences in the logic. At an intuitive level, the inferences of the logic appear to be sound. However the framework relies on a number of assumptions. Many of these are a result of deliberate omissions by the authors in order to simplify the logic:

> *... certain aspects of authentication protocols have been deliberately ignored in our treatment. In particular, while we allow for the possibility of hostile intruders, there is no attempt to deal with the authentication of an untrustworthy principal, nor to detect weaknesses of encryption schemes or unauthorized release of secrets. Rather, our study concentrates on the beliefs of trustworthy parties involved in the protocols and on the evolution of these beliefs as a consequence of communication. [?], p2.*

One assumption relates to the fact that plaintext messages (and plaintext parts of messages) are ignored. It is clear that such messages have important security implications. For example consider a protocol in which any message containing an encrypted key also contains the same key in plaintext. If we ignore the plaintext parts of messages, then the logic may claim that the protocol is "secure", despite the obvious security weakness. In other words, the logic assumes that a protocol contains no "obvious" security bugs in the form of secret information being transmitted (intentionally or unintentionally) in plaintext.

Another issue which is related to the abstraction of plaintext messages is that the protocol can not work without them. For example, an agent typically makes a nonce available to a second agent so that any message from this second agent containing the nonce is guaranteed to be recent. The first agent often makes the nonce available to the second by sending it in a plaintext message. If these kinds of messages are ignored, then no assurances can be given about whether a protocol is feasible or not. For example consider the following (unreasonable) protocol

$$\begin{array}{lll}
\text{Message 1} & A \rightarrow B: & \{N_A, A, B\}_{K_{AS}} \\
\text{Message 2} & B \rightarrow S: & \{N_A, A, B\}_{K_{AS}}, \{N_A, N_B, A, B\}_{K_{BS}} \\
\text{Message 3} & S \rightarrow B: & \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}} \\
\text{Message 4} & B \rightarrow A: & \{N_A, K_{AB}\}_{K_{AS}}
\end{array}$$

where N_A and N_B are nonces generated by A and B respectively (this is essentially just the Otway-Rees protocol [?] with the plaintext parts of messages deleted). This protocol is not feasible because B cannot extract N_A from message 1, and so there is no way for B to construct the $\{N_A, N_B, A, B\}_{K_{BS}}$ component of the second message.

Another important kind of assumption made in the logic is illustrated by the following example protocol for establishing a replacement session key (K_{AB} is assumed to be a secure session key which has already been established; the purpose of this protocol is to establish a new session key K'_{AB}):

$$
\begin{array}{lll}
\text{Message 1} & A \to B: & \{N_A\}_{K_{AB}} \\
\text{Message 2} & B \to S: & \{A, B\}_{K_{BS}} \\
\text{Message 3} & S \to B: & \{K'_{AB}\}_{K_{BS}} \\
\text{Message 4} & B \to A: & \{N_A, N_B, K'_{AB}\}_{K_{AB}} \\
\text{Message 5} & A \to B: & \{N_B, K'_{AB}\}_{K_{AB}}
\end{array}
$$

If we assume that A and B trust each other with respect to beliefs of the form $A \stackrel{K}{\longleftrightarrow} B$, then on applying the authentication logic to this protocol, we find the following final beliefs:

$$
\begin{array}{ll}
A \models A \stackrel{K'_{AB}}{\longleftrightarrow} B & B \models A \stackrel{K'_{AB}}{\longleftrightarrow} B \\
A \models B \models A \stackrel{K'_{AB}}{\longleftrightarrow} B & B \models A \models A \stackrel{K'_{AB}}{\longleftrightarrow} B
\end{array}
$$

However, the protocol is not secure since the third message can be recorded and replayed by an adversary. The problem is that, on receiving message 4, agent A sees that B has recently sent the message $A \stackrel{K'_{AB}}{\longleftrightarrow} B$ and hence A believes that B believes $A \stackrel{K'_{AB}}{\longleftrightarrow} B$. This occurs despite the fact that B did not have reason to believe $A \stackrel{K_{AB}}{\longleftrightarrow} B$ when sending message 4. Worse, in message 5, A then transmits the mistaken belief that the key is secure back to B. In other words, the validity of the logic depends on an assumption that an agent sends only messages that he/she believes[5].

Now we consider the question of whether an arbitrary valid formula is actually provable in the logic. The answer appears to be no. One reason is that there is no facility for the logic to reason negatively about the protocol, that is, to reason about the messages which, according to the protocol, will not be sent. We emphasize that an underlying assumption of the logic is that the agents involved in a protocol are faithful to the protocol with respect to the messages they sent which are relevant to the protocol. That is, the only (protocol relevant) messages they send are those specifically designated in the protocol. Consider now the following protocol (again an adaptation of the Otway-Rees protocol):

[5]This not quite accurate, since we wish to allow an agent to pass on a coded message without knowing the contents of the message. This mechanism is used for example in the Needham-Schroeder protocol.

Message 1	$A \rightarrow B:$	$A, B, N_A, \{N_A, A, B\}_{K_{AS}}$
Message 2	$B \rightarrow S:$	$\{N_A, A, B\}_{K_{AS}}, \{N_A, N_B, A, B\}_{K_{BS}}$
Message 3	$S \rightarrow B:$	$\{\{N_A, K_{AB}\}_{K_{AS}}, N_B, K_{AB}\}_{K_{BS}}$
Message 4	$B \rightarrow A:$	$\{N_A, K_{AB}\}_{K_{AS}}$

The essential difference between this protocol and the Otway-Rees protocol is that the message $\{N_A, K_{AB}\}_{K_{AS}}$ is encrypted as part of S's message to B. Hence the only way for A to receive message 4 is through B, and on receiving message 4, A is entitled to believe that (i) B has received message 3 and (ii) B has successfully decrypted message 3. Hence A is entitled to believe that B has recently seen all of the contents of message 3, and, if A believes that B trusts the server S, then A is entitled to believe that B believes that K_{AB} is a secure session key. There does not appear to be any way to carry out this kind of reasoning in the logic.

A number of the inference rules of the logic contain subtle side conditions to avoid invalid inferences. In some cases, these side conditions prevent valid inferences. For example consider the rule

$$\frac{A \models B \mid\!\sim X, \quad A \models \#(X)}{A \models B \models X}$$

A side condition on this rule is that X does not contain any encrypted messages. Now suppose that the message X contains a number of components, some encrypted and some in plaintext (including a nonce guaranteeing the freshness of the whole message). One valid inference is that A believes that B believes the plaintext parts of the message. However strictly speaking this inference is prevented by the side condition (unless we assume that the message is hierarchically assembled so that one of its subcomponents consists of just the plaintext parts and the nonce).

In summary, the Logic of Authentication framework seeks to establish a specific kind of security property – robustness with respect to the replaying of messages. It relies on many assumptions. Some of these assumptions are obvious, such as the assumption that secret information is not relayed in plaintext. Others seem to be more subtle, such as the assumption that agents only send messages which they believe. Without a suitable formalization of these assumptions, no formal guarantees can be given about the correctness of assertions proved by the logic. Also, the logic is incomplete in the sense that there are valid protocol properties which cannot be proved in the logic. It is clear, however, that the logic is a useful tool for debugging protocols.

3 Messages, Keys and Encryption

In the remainder of this paper we develop a model of possible agent interactions. One main motivation for this is to provide a justification for inferences rules such as those used in logics such as the "Logic of Authentication". In developing this model, we shall strive for generality, particularly in the notion of time which is incorporated. We begin with the basic notions of message and encryption.

There are important interactions between a protocol and the encryption scheme employed. In particular, if an encryption scheme is amenable to a probabilistic analysis, then it is desirable to extend such a probabalistic analysis to any protocol based on the scheme. However, such an analysis is likely to be dependent on the specific properties of the encryption scheme considered. In this paper we choose to consider a simpler and arguably more fundamental notion: security properties which are independent of the underlying message representation and encryption scheme. In essence we assume an idealized model of message representation and encryption. First, 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.

These assumptions are realized as follows. Let $\mathcal{T}$ denote a set of *atomic* tokens[6]. This set will represent the keys, the nonces, and the non-decomposable message of the model. Notationally, we shall use K to denote tokens which are used as keys and N to denote those used as nonces. The domain $\mathcal{M}$ of messages is defined to be either

- an atomic token from $\mathcal{T}$;
- $\{M\}_K$, the *encryption* of message M with key $K \in \mathcal{T}$, or
- $(M_1, \cdots, M_n)$, the *tupling* of messages $M_1, \cdots, M_n$, $n \geq 2$,

where M, $M_1, \cdots, M_n$ range over messages in $\mathcal{M}$.

Now, given a set of message S, an agent has a limited ability to decrypt and de-tuple these messages into their constituents and also built up new messages by encryption and tupling. Specifically, define $S^\star$, the set of messages deducible from

[6]We will assume that $\mathcal{T}$ contains tokens for naming agents, so that if A is an agent, then we will sometimes treat A as a message (or message component) which can be sent and received.

S, to be either (i) any element of S, (ii)[7] M such that $\{M\}_K$ and K are in $S^\star$, (iii) M such that $(\cdots, M, \cdots)$ is in $S^\star$, (iv) $\{M\}_K$ such that M and K are in $S^\star$, and (v) $(M_1, \cdots, M_k)$ such that each M_i is in $S^\star$. This construction is essentially a free algebra construction over the generators $\mathcal{T}$, $(\_, \cdots, \_)$ and $\{\_\}_\_$, with auxiliary operations, call them $decrypt\_(\_)$ and $proj_i(\_)$, $i \geq 1$, which satisfy the following equations:

$$proj_i(M_1, \cdots, M_n) = M_i, \quad 1 \leq i \leq n$$
$$decrypt_K(\{M\}_K) = M$$

4 Traces and Some Basic Assumptions

Let $\mathcal{A}$ be the set of all agents of interest (that is, $\mathcal{A}$ includes not only the principals of the protocol at hand, but also any adversaries). Agents interact by sending and receiving of messages; a trace of these message *sends* and *receives* serves to record the interaction. Define an *event* to be either of the form $send(M)$, indicating that the message M is sent, or $receive(M)$, indicating the message M is received, or $new(M)$, indicating the generation of the *basic* message M (M may be either a nonce or a key), or *tick*, indicating an event corresponding to a change in an agent's internal state. The first two kinds of events record agent interactions; the last two events are essentially for bookkeeping purposes. Events are subscripted if necessary to indicated different occurrences of the same event. This is the case when the same message is received by a number of different agents, or when a message is replayed by an adversary and is received by the same agent a number of times.

Now, a record of agent interactions must include not only the set of events for each agent, but also the order of these events. This can be achieved by assuming that some global clock associates a time to each message send/receive event. However committing to such a specific notion of time may lead to a model which is not applicable to systems where agents have unsynchronized clocks, or where non-standard notions of time such as Lamport clocks [?] are employed. We therefore strive for a definition of model that makes only minimal assumptions about the notion of time. One such assumption is that each agent has a notion of "before" and "after" which defines a total pre-order on the events that the agent perceives. Hence

[7] 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 $S^\star$.

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 pre-order.* □

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 pre-order on T_A is denoted by $\leq_A$. We write $e \equiv_A e'$ if $e \leq_A e'$ and $e' \leq_A e$. We write $e <_A e'$, and say that e *precedes* e', if $e \leq_A e'$ but $e \not\equiv e'$. The subscripts A on $\leq$, $\equiv$ and $<$ shall be dropped when they are clear from context. The use of an ordered set to provide an abstract treatment of time in the definition of trace bears a superficial similarly to the treatment of time in the model theory of temporal logic [?, ?, ?]. However our subsequent development and use of traces bears little relationship to work in that area.

We remark that an agent typically sends a message to some designated recipient. However, since the communication medium is assumed to be insecure, any message sent is essentially broadcast to the world. This means that any unencoded information which designates the intended recipient of a message has negligible effect on security; we therefore choose to ignore recipient information. Further, we make no assumptions about the correct behavior of the network – messages may be lost or even duplicated due to network failures and errors.

There are many factors which constrain the messages that an agent can send. First, it is clear that if a message M is received by an agent, then some agent must have previously sent M. This constraint can be imposed by requiring that 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 transmission.

Definition 2 (Serializable Traces) *A trace T is* serializable *if there is a total pre-order on T which conservatively extends the pre-orders $\leq_A$, $A \in \mathcal{A}$, and is such that each event receive(M) is a preceded by an event send(M).* □

A further assumption is that between any two events, there lie only a finite number of events. This can be justified on physical grounds; it is also necessary to eliminate pathological traces.

Definition 3 (Boundedness) *A trace T is* bounded *if, for all agents A, every sequence of distinct events of the form $e_1 \leq_A e_2 \leq_A \cdots \leq_A e'$ is finite.* □

Note that boundedness also implies that given any event e, there is a finite set of immediate predecessors of e, denoted $pred(e)$, and that any two elements e_1 and

e_2 of $pred(e)$ are such that $e_1 \equiv e_2$. The final assumption of this section is that each event of the form $new(M)$ generates a new basic messages.

Definition 4 (Message Generation) *A trace T respects message generation if for all distinct events of the form $new(M_1)$ and $new(M_2)$, the messages M_1 and M_2 are distinct.* $\square$

In what follows, we use the term trace to mean a trace which is serializable, bounded and respects mesage generation. One important assumption that has not been discussed is that messages cannot be "guessed". An agent can only send messages which have been generated by the agent or have been received in other messages. This is somewhat complicated by the fact that an agent starts with some set of shared secrets and keys, and depends on some definitions presented in the next section. A statement of this assumption is therefore deferred until the next section, where it appears as part of the definition of model (definition **??**).

5 Protocols

In the previous section, we defined the basic notion of a trace of the messages sent and received by each agent. This essentially models the situation where agents are free to send and receive messages at random. We now apply the additional restriction that some designated subset of the agents adhere to some protocol, call it P, for message sending and receiving. That is, any agents in this designated set (call these agents the *principals* of P) are assumed to send and respond to messages in the manner outlined by P. Now, protocols are typically specified by giving a template for message sends and receives. For example the following describes an adaptation of the Otway-Rees protocol.

$$\begin{array}{lll}
\text{Message 1} & A \rightarrow B: & A, B, \{N_A, B\}_{K_A} \\
\text{Message 2} & B \rightarrow S: & \{N_A, B\}_{K_A}, \{N_B, A\}_{K_B} \\
\text{Message 3} & S \rightarrow B: & \{N_A, B, K_{AB}\}_{K_A}, \{N_B, A, K_{AB}\}_{K_B} \\
\text{Message 4} & B \rightarrow A: & \{N_A, B, K_{AB}\}_{K_A}
\end{array}$$

A trace which is faithful to this protocol is essentially a trace whose messages follow the protocol. 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 messages 1, 2, 3 and 4, but rather a specification that:

- to establish a session key with B, A should sent message 1;
- if B receives message 1, then B should respond with message 2;
- if the server S receives message 2, then S should send out a new secure key;
- if B receives message 3, then B should accept the new key as a secure key for A-B conversations and also repond with message 4, and
- if A receives message 4, then the protocol is complete and A should use the new key for conversations with B.

Such a specification requires reasoning about an the beliefs of each agent. Let a *belief* be either of the form $shared_S(M)$ where S is a set of agents and M is a message, or of the form $fresh(N)$ where N is a nonce. The first indicates the belief that M is a shared secret between the agents in S. The second indicates that the nonce N has been recently generated.

Also note that the variables A, B, N_A, N_B, K_A, K_B and K_{AB} appearing in the protocol may denote specific agents, messages or keys, or they may be parameters which can be instantiated by different keys to give instances of the protocol. To distinguish between these two cases, we use the (possibly subscripted) variables α, η, κ and μ to respectively denote parameters over agents, nonces, keys and messages. A *message template* is either a message, a message parameter or of the form $\{x\}_y$ where x is a message template and y is a key, key parameter, or of the form $(x_1, \cdots, x_k)$ where the x_i are message templates. A *condition* is either of the form $shared_S(x)$, $fresh(y)$ or $receive(x)$, where S is a set of agents or agent parameters, x is a message template and y is a nonce or a nonce parameter. A *response* is either of the form $send(x)$ or $shared_S(x)$. Finally

Definition 5 (Protocol) *A* <u>protocol</u> *P consists of a set of principals and a mapping from each principal A to a collection P_A of (parameterized) rules of the form $C \Longrightarrow R$, where C is a set of conditions and R is a set of responses.* $\square$

The condition part of a protocol rule represents a set of preconditions that an agent tests before applying a rule. For example a belief $shared_S(x)$ represents the precondition that the agent believes that x is known only to the agents in S i.e. x is an S shared secret. The variant of the Otway-Rees protocol described above is represented by the following collection of protocol rules:

A: $\mathit{shared}_{A,S}(\mathcal{K}),\ \ \mathit{fresh}(\nu) \Longrightarrow \mathit{send}(A, \alpha, \{\nu, \alpha\}_{\kappa})$
$\quad \mathit{shared}_{A,S}(\mathcal{K}),\ \ \mathit{receive}\,(\{\nu, \alpha, \mathcal{K}'\}_{\kappa}),\ \ \mathit{fresh}(\nu) \Longrightarrow \mathit{shared}_{A,\alpha,S}(\mathcal{K})'$

B: $\mathit{shared}_{B,S}(\mathcal{K}),\ \ \mathit{receive}(\alpha, B, \mu),\ \ \mathit{fresh}(\nu) \ \ \Longrightarrow \ \ \mathit{send}\,(\mu, \{\nu, \alpha\}_{\kappa})$
$\quad \mathit{shared}_{B,S}(\mathcal{K}),\ \ \mathit{receive}\,(\mu, \{\nu, A, \mathcal{K}'\}_{\kappa}),\ \ \mathit{fresh}(\nu) \ \ \Longrightarrow \ \ \mathit{send}(\mu),\ \ \mathit{shared}_{\alpha,B,S}(\mathcal{K}')$

S: $\mathit{shared}_{S,\alpha_1}(\mathcal{K}_1),\ \ \mathit{shared}_{S,\alpha_2}(\mathcal{K}_2),\ \ \mathit{receive}\,(\{\nu_1, \alpha_1\}_{\kappa_1}, \{\nu_2, \alpha_2\}_{\kappa_2}),\ \ \mathit{shared}_{S,\alpha_1,\alpha_2}(\mathcal{K})$
$\quad \Longrightarrow \ \ \mathit{send}\,(\{\nu_1, \alpha_1, \mathcal{K}\}_{\kappa_1}, \{\nu_2, \alpha_2, \mathcal{K}\}_{\kappa_2})$

An *instance* of a protocol rule r is defined by a substitution θ which maps all of the parameters of the rule in the obvious manner: agent parameters are mapped into elements of $\mathcal{A}$, nonce and key parameters are mapped into elements of $\mathcal{B}$ and message parameters are mapped into elements of $\mathcal{M}$. We write $r\theta$ to denote the rule obtained by replacing the parameters in r according to θ, and we say that $r\theta$ is an instance of r. Note that this implicitly defines the scope of a parameter to be the protocol rule in which it occurs; parameters in different rules are independent. Protocol rule parameters could alternatively be treated by introducing a universal quantification construction at the front of protocol rules.

In essence, a trace T adheres to a protocol if the trace consists of instances of protocol rules. However, a protocol rule can only be applied when the conditions of the rule are satisfied. This requires reasoning about the set of beliefs held by each agent A during the trace. In essence the definition of model consists of a trace and accompanying machinery to reason about these sets of beliefs. Before giving this definition we need some preliminaries. These definitions are all in the context of a trace T and protocol P.

Initially each agent is given some set of beliefs about shared secrets and keys. Now, as each event passes, this initial set of beliefs may increase by the establishment of new beliefs though the use of instances of the protocol rules, and it may decrease because the agent discards a belief. To record these changes, we define a *belief function* to be an $\mathcal{A}$-indexed collection of functions, denoted *beliefs*, such that the function corresponding to an agent A, denoted $\mathit{beliefs}_A$, is a mapping from T_A to sets of beliefs, and this function respects $\equiv$ (that is, $e \equiv e'$ implies $\mathit{beliefs}_A(e) = \mathit{beliefs}_A(e')$). We say that b *is a belief of* A if $b \in \mathit{beliefs}_A(e)$ for some event $e \in T_A$. Given a belief mapping *beliefs*, the initial beliefs of each agent can be defined as those beliefs which are held at some event e and at all events preceding e:

$$initial_A(beliefs) = \{b : (\exists e)(\forall e' \le e)\ b \in beliefs_A(e')\}$$

Now, given a trace and a belief function *beliefs*, we can define the set of messages known by an agent A at each event e:

$$known_A(e) \quad = \quad \left\{ M : \begin{array}{c} receive(M) \text{ precedes } e \text{ in } T_A, \text{ or} \\ new(M) \text{ precedes } e \text{ in } T_A, \text{ or} \\ shared_S(M) \text{ is in } initial_A(beliefs) \end{array} \right\}^{\star}$$

As mentioned at the end of the previous section, an agent can only make reference to known messages. Therefore define that a response $send(M)$ or $shared_S(M)$ is *known* at event e if $M \in known_A(e)$. For a protocol principal, any response must be according to the protocol. Therefore define that a response r is *protocol enabled* at event e if there is an instance of a protocol rule of the form $C \Rightarrow R$ such that

- for all beliefs $b \in C$, $b \in beliefs_A(pred(e))$[8];
- for all conditions $receive(M)$ in C, $receive(M)$ is a preceding event in T_A, and
- r appears in R.

Finally, for a protocol principal, define that a response is *enabled* if it is known and protocol enabled, and for any other agent, a response is *enabled* if it is known. In other words, principals must adhere to the protocol, whereas adversaries can be more creative. We can now define:

Definition 6 (Model) *A* <u>model</u> *for a protocol P is a pair $(T, beliefs)$ where T is a trace and beliefs is a belief function, which satisfy the following conditions, for all agents $A \in \mathcal{A}$:*

1. *If $b \in beliefs_A(e)$ then either*
   (a) *b is in $beliefs_A(pred(e))$;*
   (b) *b is enabled at e, or*
   (c) *b is either $shared_S(N)$ or $fresh(N)$ where $new(N)$ is an event in $pred(e)$.*

[8] This denotes $beliefs_A(e')$ for some e' in $pred(e)$. Since the elements in $pred(e)$ are equivalent modulo $\equiv$, it does not matter which e' is chosen in $pred(e)$.

2. *If* $fresh(N) \in beliefs_A(e)$ *then there is an event* $new(N)$ *preceding* e.

3. *If* e *is an event of the form* $send(M)$ *then it is enabled at* e. $\square$

Condition 1 states that the beliefs of an agent at any event must either be (a) a belief held previously, (b) a belief established at an immediately preceding event, or (c) a belief established as a result of generating a new message. Condition 2 states that the freshness beliefs can only be generated through the generation of a nonce. Condition 3 states that message sends must be known, and if the agent is a protocol principal, then the message send must also be in accordance with the protocol.

Note that an agent may discard beliefs. However, there is no mechanism for an agent A to hold the belief $shared_M\{A\}$ and then share the secret M with another agent B and then update its belief to $shared_M\{A, B\}$. The ability to update beliefs in this way would complicate the model by requiring that the application of protocol rules be atomic. For example consider a protocol rule whose response set includes sending a message to share the secret M with agent B and an action to update the belief set to reflect the sharing of this secret. Now if the secret is shared before the agent updates its beliefs, then an inconsistent state would result. Not only would the ability to update belief sets complicate the model, but such an ability appears to be unnecessary, since, in the above example, A's initial belief about M could be changed to $shared_M\{A, B\}$. In general, this approach means that beliefs about sharing are of the form $shared_M\{A_1, \cdots, A_n\}$ where $A_1, \cdots, A_n$ include *all* of the agents that will ever share M. This appears to be desirable because it yields a certain level of modularity. Further, it enables the security properties of a model to be defined without reference to synchronization between each agent's clocks. In fact the truth of a belief can be described in terms of universal validity, that is, validity everywhere the model.

Definition 7 (Valid Beliefs) *A belief* b *is* <u>valid</u> *in a model* $(T, beliefs)$ *if either it is of the form* $fresh(M)$, *or of the form* $shared_S(M)$ *such that if* $M \in known_A(e)$ *for some event* e, *then* $A \in S$. $\square$

In other words a belief about sharing a message is valid if the only agents who know about the message are those identified in the sharing belief. Note that beliefs about nonces are always valid. Furthermore, beliefs about nonces cannot be shared by agents — the only way for an agent to create such beliefs is by generating a nonce. Hence there is a sense in which nonces should be considered in a category separate from beliefs. However we choose to do so for convenience, since in the next section when we consider the expiration of beliefs, the same mechanism can be applied to both nonce beliefs and sharing beliefs.

We can now define a notion of security which is independent of time considerations. In essence this notion is that secrets are not compromised. Specifically, it is the condition that the beliefs of agents about secrets are actually valid. However, since the definition of model makes no assumptions about the validity of the initial beliefs of agents, the following definition must be phrased in terms of preserving validity. First, define that a model $(T, \mathit{beliefs})$ is *initially valid* if, for all beliefs $\mathit{shared}_S(M)$ in $\mathit{initial}_A$, it is the case that $M \in \{M : \mathit{shared}_S(M) \in \mathit{initial}_B\}^\star$ implies $B \in S$.

Definition 8 (Semi-Secure) *A protocol P is <u>semi-secure</u> if in all initially secure models, all beliefs held by a principal A are universally valid.* □

We reserve the term "secure" for the more involved notion of security dealing with questions such as: can an adversary replay old messages to re-establish a stale keys or belief? To define this notion of security requires a more detailed discussion of time, and in particular, the freshness of messages, and the expiration of nonces and beliefs. This is the subject of the next section.

6 Beliefs and Time

The treatment of time is one of the most important aspect 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 pre-determined 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 $t + \delta$. More generally, δ may vary from nonce to nonce. Instead of modeling the behavior of nonces by committing to such a specific mechanism, a more abstract characterization is used here. This is built on a minimal assumption about nonces: each nonce eventually expires. Specifically, let A be an agent, $e \in T_A$ and $b \in \mathit{beliefs}_A(e)$ and define that that the belief b *expires* if there exists an event $e' >_A e$ such that $b \notin \mathit{beliefs}_A(e)$. Now, the assumption about nonce expiry can be stated as a condition on a model $(T, \mathit{beliefs})$ as follows:

All beliefs of the form $\mathit{fresh}(N)$ in $\mathit{beliefs}_A(e)$ expire. (6.1)

where A ranges over all agents, and e ranges over all events in T_A. Since the only mechanism for an agent to add $\mathit{fresh}(\eta)$ to its set of beliefs is by generating η, condition **??** insures that the nonce eventually expires and is never again considered fresh.

Now consider the beliefs held by agents. As for nonces, an important part 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 another later time. An appropriate condition on expiry of beliefs is therefore:

All beliefs of the form $\mathit{shared}_S(M)$ in $\mathit{beliefs}_A(e)$
either expire or are enabled at some event $e' >_A e$. (6.2)

This means that the only way for an agent to maintain a belief indefinitely is for the belief to be enabled indefinitely. We can now define:

Definition 9 (Security) *A model for a protocol P is a <u>timed model</u> if it satisfies* **??** *and* **??**. *A timed model is <u>secure</u> if it is semi-secure and for all beliefs b of a principal A there is an event e such that $e' >_A e$ implies $b \notin \mathit{beliefs}_A(e')$. A protocol P is <u>secure</u> if each initially secure timed model for P is secure.* $\square$

We now discuss why this definition captures an important notion of security. The most important failure mode of a protocol is where an adversary replays sequences of old messages to convince a principal of the validity of a belief. However, in certain circumstances, this may be not be considered insecure. For example suppose an agent A sends a message M to another agent B which is intercepted by an adversary Z and does not reach B. Then suppose that 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 the 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. 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.

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[9]. 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 the essence of what it means for a protocol to be secure. In the appendix we support this claim by considering two different models which incorporate more specific notions of time than our definition, discuss possible definitions of security in the contexts of these models, and then provide comparisons of these notions of security with our definition.

We conclude this section by proving that a number of security properties are undecidable.

Theorem 1 *Given a protocol P, the following questions are not decidable:*
(a) Is P semi-secure?
(b) Is P secure?
(c) Is P secure if it is semi-secure?

Proof: A protocol can essentially be viewed as a rewriting system. In fact it is easy to code Post's correspondence problem as a protocol such that a belief of the form

[9]Note that the notion of semi-security is not at issue since it does not rely in any notion of time. We shall assume that regardless of the notion of time involved, any definition of security (of a model or protocol) includes semi-security.

$shared_{\{A\}}(M)$ is held by an agent A iff 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 iff the correspondence problem has a solution. This outline proves (a). The remaining parts can be proved by similar methods. $\square$

7 Future Work: Model Checking

Although the general problem of determining protocol security is undecidable, progress has been made on the automatic verification of important subclasses of protocols. Broadly speaking, there are two approaches here — proof theoretic and model theoretic. In the former approach, a set of inference rules is constructed for reasoning about the protocol. Then the protocol is evaluated by considering the judgments derivable from these inference rules. The difficulty here is in finding an appropriate set of inference rules.

Model checking, on the other hand, is a more direct approach — essentially we evaluate the protocol by constructing models and checking that these models are secure. Clearly we cannot check all models of a protocol since there are an infinite number of models, and these models model may be infinite. Hence a model checking approach can only be used if there is some way to limit the number of models which must be constructed. Clearly this cannot be done in general (see Theorem **??**); however

Our ongoing work into model checking suggests that for a reasonable class of protocols, including most of the protocols from the literature, it is possible to verify security properties by model checking. The approach involves reducing the problem of checking all models to just checking one "canonical" model. Although this canonical model is infinite, it only involves a finite number of nonces and keys. This enables us to represent the sets of messages that each agent has received, knows and believes, and the inter-relationships between these sets, using the set calculus of [?]. In essence, the fact that there are only finite number of keys means that encryption can be represented by function symbols and decryption can be represented by projection over these function symbols. For example the encryption of a message M with key K can be written as $K(M)$, and the decryption of a message M using key K can be written as $K^{-1}(M)$. Importantly, $K^{-1}(K'(M))$ is equal to M only if $K = K'$. Constraints can be used to model the relationships between sets of messages. For example if $Known_A$ denotes the set of message known by an agent A then consider the constraint

$$Known_A \quad \supseteq \quad p(Known_A, Known_A) \quad \cup \quad K(Known_A).$$

This recursive constraint says that $Known_A$ is a superset of two quantities. The first quantity consists of all those messages obtained by pairing any messages from $Known_A$ (the function symbol p is used here to represent pairing). The second quantity consists of all messages obtained by encrypting a message from $Known_A$ with the key K. However this last quantity is not quite correct, because $Known_A$ should only include these messages if the key K is in $Known_A$. This can be corrected by instead writing $p^{-1}_{(2)}(p(K(Known_A), K \cap Known_A))$, where $p^{-1}_{(2)}$ denotes projection on p at the second place (for example $p^{-1}_{(2)}(\{p(1,2), p(3,4)\})$ is $\{2,4\}$), and K is used here both as a constant as well as a unary symbol for represented encryption with the key K. This new expression is equal to $K(Known_A)$ if $K \in Known_A$ and is equal to $\{\}$ otherwise. Once set constants have been constructed corresponding to the canonical model, an algorithm in [?] can be used to construct their least solution, and from this it is possible to immediately determine whether the protocol is semi-secure.

In conclusion, there is strong evidence that, for protocols of practical importance, it is possible to use a model checking approach to verify security properties.

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] S. M. Bellovin and M. Merritt, "Limitations of the the Kerberos Authentication System", *Computer Communication Review*, Vol. 20, No. 5, pp. 119-132, October 1990.

[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] E. Biham and A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems*, Weizmann Institute of Science Technical Report CS90-16, July 1990.

[5] E. Biham and A. Shamir, *Differential Cryptanalysis of the Full 16-round DES*, Technion Computer Science Department Technical Report 708, December 1991.

[6] 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.

[7] M. Burrows, M. Abadi and R. Needham, "A Logic of Authentication", Research Report No. 39, DEC SRC, 48 pages, 1989.

[8] 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.

[9] U. Feige, A. Fiat, and A. Shamir, "Zero Knowledge Proofs of Identity", *Proceedings* 19^{th} *ACM Symposium on the Theory of Computing*, pp. 210-217, May 1987.

[10] J. Y. Halpern and R. Fagin, "A Formal Model of Knowledge, Action and Communication in Distributed Systems", *Proceedings* 4^{th} *ACM Symposium on Principles of Distributed Computing*, pp. 224-236, August 1985.

[11] 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.

[12] N. C. 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.

[13] 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

[14] 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.

[15] C. A. Meadows, "Applying Formal Methods to the Analysis of a Key Management Protocol", Technical Report No. 9265, Naval Research Laboratory, Washington DC., September 1990.

[16] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", *CACM* Vol. 21, No. 12, pp. 993-999, December 1978.

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

[18] J. Plumstead, "Inferring a Sequence Generated by a Linear Congruence", *Proceeding* 23^{rd} *IEEE Symposium on the Foundation of Computer Science*, pp. 153-159, October 1982.

[19] J. D. Tygar and B. S. Yee, "Strongbox: A System for Self-Securing Programs", *CMU Computer Science: A* 25^{th} *Anniversary Commemorative*, R. Rashid Ed., ACM Press and Addison-Wesley, pp. 163-197, 1991.

Appendix: Variations of the Security Definitions

In this appendix we consider two important variations on the definition of model. The aim of this section is to justify the abstract nature of our definitions by showing that different and more concrete notions of model do not lead to different notions of security. The first modification we shall consider involves restriction the definition of model so that the expiry of beliefs respects their order of generation. We now elaborate. Let $(T, \mathit{beliefs})$ be a model of a protocol P and let A be an agent. An *A-segment* is a subset of the events T_A such that if e_1 and e_2 are in E and $e_1 \leq_A e \leq_A e_2$ then $e \in E$. An *A-lifetime* L of a belief is a maximal segment of events such that the belief is held by A at each event in L. Note that a belief may have several A-lifetimes. A model $(T, \mathit{beliefs})$ is *fair* if for each protocol principal A, if 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 their order of generation in the sense that if a belief b is established before b' then b will expire before b' (although beliefs may be re-established at some later time).

Now, consider defining protocol security in terms of security over all fair models (instead of over *all* models). Such a definition is equivalent to the original definition of security. The reason is that any insecure model for a protocol P can be transformed into a insecure fair model by extending the life of various nonces and beliefs so that the fairness condition is satisfied. As another alternative definition of security, consider the more restrictive definition which specifies that, in all fair models, each agent's beliefs has only one lifetime. It turns out that such a definition (call it *single lifetime security*) is somewhat restrictive. We now discuss why this is so, and establish a connection between this restrictive notion of security and our previous definition.

We first define the notion of the path by which a belief is established. Essentially this records the events leading up to the establishment of a belief in the form of a tree of protocol rule instances. Such a record is a witness to an agent's belief, and for this reason shall be referred to as a proof[10].

Definition 10 *A proof p of a belief b is defined inductively as follows. If b is of the form $\mathit{fresh}(N)$ or else $b \in \mathrm{initial}_A$, then $\rightarrow b$ is a proof of b. If b is of the form $\mathit{shared}_S(M)$ such that there is an instance of a protocol rule $C \rightarrow R$ where $b \in R$ then $C' \rightarrow b$ is a proof of b where C' is the result of replacing each occurrence of a belief in C by a proof for that belief.* $\square$

[10] Taking the analogy further, each protocol rule can be viewed as a (somewhat limited) rule of inference.

For example, referring back to the Otway-Rees protocol, one proof for B's belief $shared_{A,B,S}(K)$ is

$$shared_{B,S}(K),\ \ receive(M, \{N, A, K'\}_K),\ \ fresh(N)\ \ \rightarrow\ \ shared_{A,B,S}(K')$$

Proofs which contain other proofs arise when a belief which is not an initial belief is used by an agent to establish another belief. The Otway-Rees protocol does not exhibit such behavior. A proof $C \rightarrow b$ is *established* at an event e if b is a currently held belief and if all proofs $p \in C$ are held at $pred(e)$. Such a proof is said to establish b.

Now, the reason why single lifetime security is very restrictive is that it does not allow for the possibility that there may be more than one proof by which an agent can establish a belief. For example consider the Otway-Rees protocol. Supposing that S reuses a key for $A - B$ conversations (for example, suppose that A requests a second conversation with B before S considers the key to have expired)[11], there could be two different proofs by which A could establish $shared_{A,B,S}(K)$ for some key K. By limiting consideration to the case where each belief has only one proof, we have the following connection between security and single lifetime security.

Proposition 1 *Let P be a secure protocol. Let $(T, beliefs)$ be a model which fair. Then for each principal A and for each belief b, if there is only one proof which establishes b, then b has at most one A-lifetime.*

Proof: We show that if b is only established by one proof, call it p_b, then if b has two distinct A-lifetimes, call them L_1 and L_2, then the protocol must be insecure. This is proved by structural induction on p_b. In the base case of a proof of the form $\rightarrow p_b$, the proof is easy, since in this case no protocol rules are used to establish b, and so either b is an initial belief (which eventually expires, but can never be re-established), or else b is of the form $fresh(N)$ which again can never be re-established. Hence in either case, b can only have one lifetime.

In the induction case, note that there must be some "first" event e_b at which the belief b holds. Otherwise $b \in initial_A$, and then there would be a proof of the form $\rightarrow p_b$ for p_b. Without loss of generality, suppose that $e_b \in L_1$. Now, let p_b be of the form $C \rightarrow p_b$. Clearly p_b must be enabled at e_b. Suppose that C does not contain a proof. This means that C only contains elements of the form $shared_S(M)$, and

[11] In certain circumstances we may wish to prevent this kind of behavior. This could be achieved by modifying the definition of model to introducing a special kind of key/nonce generation primitive which generates atomic new tokens which can only be used once.

so p_b is enabled at all events $e \geq e_b$. It is therefore easy to construct a timed-model which is not secure, and this would imply that the protocol P was not secure.

Hence it must be the case that C contains a proof. Let $p \in C$ and write p as $C' \rightarrow b'$. Since the proof p_b is enabled at e_b, the proof p must be established at $pred(e)$. Hence the belief b' holds at $pred(e_b)$. Moreover, by the induction hypothesis, the belief b' has only one lifetime, say $L_{b'}$. Therefore $pred(e) \in L_{b'}$, and since L_1 is not strictly contained in $L_{b'}$, it must be the case that there exits an event e' in $L_{b'}$ such that b' does not hold at e'. Since b' has only one lifetime, this means that b' does not hold for any event after e'. Since L_1 and L_2 are not distinct and are both maximal intervals, there must exist an event $e_{1,2}$ which is later than all events in L_1 and earlier than all events in L_2. Clearly $e_{1,2} > e'$. Now the belief b' is not held by A at e' or at any later event, and so the proof p cannot be established at $e_{1,2}$ or at any later events, including all those events in L_2. It follows that L_2 cannot be a lifetime of b. $\square$

We now consider a second modification to the definition of model. Suppose that each agent is equipped with a clock and that these clocks are perfectly synchronized. Each event will be timestamped (with a real number) according to the time that it occurred. Further suppose that each agent only holds a belief for some fixed time δ. Specifically, suppose that each agent only holds a belief at some time t if it has been enabled during the period δ before t. Call such models *global clock models.*

As before, if security is defined in terms of security in these new models, the new definition of security is equivalent to the original definition. This is because an insecure model can be modified to become an insecure global clock model. We now address the issue of whether, in the context of global clock models, a secure protocol yields any additional properties or guarantees. For example, in the original definition an agent A is assured that if a secret is shared with another agent B, then B will eventually consider the secret to be stale, but no bound is placed on the time for this to happen.

We conclude this section with some observations on global clock models. Security, as we have defined, is only concerned with the eventual expiration of beliefs and nones. Even though the two notions of security obtained by considering our original model and the global clock model are equivalent, we can still raise the question of whether anything about the expiration beliefs can be said beyond simply asserting that they will "eventually expire". For example we might expect that A can be assured that a shared secret will be considered stale by B within a time bound such as some fixed multiple of δ. However this is not the case. To see this, first define, for an agent A, the *extent* of a belief to be $t_{last} - t_{first}$ where t_{first} is the first time and t_{last} is the last time that A holds the belief. Now consider the following protocol whose only principal is A:

$$\begin{array}{lrcl}
\text{A:} & \mathit{fresh}(\kappa) & \Longrightarrow & \mathit{shared}_A(\kappa) \\
& \mathit{shared}_A(\kappa) & \Longrightarrow & \mathit{shared}_A(\kappa, \kappa) \\
& \mathit{shared}_A(\kappa, \kappa) & \Longrightarrow & \mathit{shared}_A(\kappa, \kappa, \kappa) \\
& \vdots & \vdots & \vdots \\
& \mathit{shared}_A(\kappa^{n-1}) & \Longrightarrow & \mathit{shared}_A(\kappa^n)
\end{array}$$

where κ^n denotes the n-tuple $(\kappa, \ldots, \kappa)$. We can easily see that this protocol is secure. Moreover, it is possible to construct a global clock model of this protocol such that there is a belief with an extent of $n\delta$. We conjecture that in general a protocol is secure iff in all global clock models all beliefs have extents bounded by $(n+1)\delta$, where n is the number of rules in the protocol at hand.

In summary then, we see that major modifications to the treatment of time in our original definition of model do not change the notion of security that arises.