A Semantics for a Logic of Authentication
(Extended Abstract)

Martín Abadi
Digital Equipment Corporation
Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301
ma@src.dec.com

Mark R. Tuttle
Digital Equipment Corporation
Cambridge Research Lab
One Kendall Square, Bldg. 700
Cambridge, MA 02139
tuttle@crl.dec.com

Abstract: Burrows, Abadi, and Needham have proposed a logic for the analysis of authentication protocols. It is a logic of belief, with special constructs for expressing some of the central concepts used in authentication. The logic has revealed many subtleties and serious errors in published protocols. Unfortunately, it has also created some confusion.

In this paper, we provide a new semantics for the logic, our attempt to clarify its meaning. In the search for a sound semantics, we have identified many sources of the past confusion. Identifying these sources has helped us improve the logic's syntax and inference rules, and extend its applicability. One of the greatest differences between our semantics and the original semantics is our treatment of belief as a form of resource-bounded, defeasible knowledge.

1 Introduction

Authentication is the act of determining the identity of a principal (such as a person, computer, or server) in a computer system. Authentication usually plays an important role in secure systems, since a principal controlling a resource must have some way of identifying principals requesting access to the resource. Authentication typically depends on secrets, such as passwords or encryption keys, that one principal can reveal or somehow use to prove its identity to others. Before these secrets can be used, however, they must be distributed to the principals in some way. An *authentication protocol* is a description of how these secrets are distributed to principals, and how these secrets are used to determine principals' identities.

This paper will appear in *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, Montreal, Canada, August, 1991.

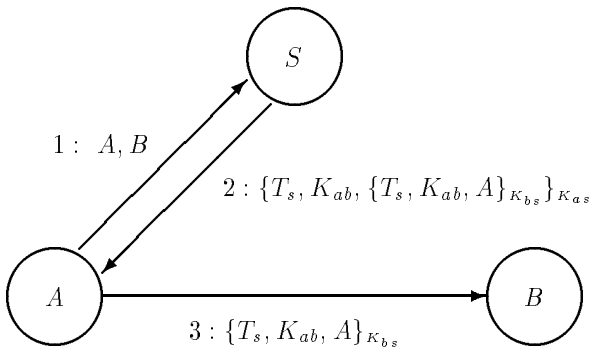

Figure 1: An authentication protocol.

A simple authentication protocol is given as an example in Figure 1. (This is actually a very incomplete description of the Kerberos key distribution protocol [MNSS87, KNS90].) The three principals involved are a server S trusted to generate good encryption keys, and two principals A and B. The goal of this protocol is for A and B to acquire a key that they can use in their communication. Principal A begins by sending a request for a key to the server S. The server responds with a message containing a timestamp T_s, the new key K_{ab} for A and B, and an encrypted submessage. The entire message is encrypted with the key K_{as} known only to A and S, so after decrypting the message, A believes the message must have come from S. Moreover, since A trusts S to generate good keys, A believes that K_{ab} was a good key when S sent the message. Since the message contains a recent timestamp T_s, A believes that S sent the message recently, and hence that K_{ab} is still a good key for use with B. Principal A then forwards to B the encrypted submessage it received in the message from S. This submessage contains the timestamp T_s, the key K_{ab}, and A's name, all encrypted with the key K_{bs} known only to B and S. Consequently, B—just like A—believes the message was recently con-

structed by S, and hence that K_{ab} is a good key to use with A.

This informal line of reasoning suggests one possible specification satisfied by the protocol: if A and B initially believe that K_{as} and K_{bs}, respectively, are good keys for use with S, then they will believe that K_{ab} is a good key for use with each other at the end of the protocol's execution. Informal arguments about authentication protocols, however, are notoriously error-prone, in part because their correctness is dependent on subtle assumptions being made about the system. Furthermore, a great number of protocols have been proposed, and many of them differ from each other in subtle ways, both in the assumptions they make and in the specifications they satisfy.

Burrows, Abadi, and Needham [BAN89] have proposed a logic specifically tailored to the analysis of authentication protocols. The purpose of the logic is to formalize our informal reasoning about authentication protocols, and to explain these protocols and their differences. The logic provides a language for describing the beliefs of the parties involved in a protocol, and a set of inference rules that describe how these beliefs evolve as the result of communication. It includes special constructs for expressing many of the central concepts used in authentication. Some of these concepts are illustrated in the protocol above: *good keys* are used to determine who has sent the various messages, a *trusted authority* (the server) is trusted to generate good encryption keys, and timestamps are used to prove that messages are *fresh*, meaning that they have not been sent before the start of the current authentication. Many important aspects of authentication are deliberately ignored by the logic. For example, it sheds no light on the secrecy of message contents, but it does explain the beliefs a message recipient has about the identity of the sender.

To use the logic to analyze a protocol, one first writes down a list of formulas describing the assumptions being made about the system, annotates the protocol with formulas holding at various points in the protocol's execution, and uses the inference rules to derive the formula stating the protocol's expected correctness condition. One's inability to do so often indicates an error in the protocol. In part because the logic has helped uncover serious errors in published protocols [BAN89, Rac89], it has received a great deal of attention [ABKL90, AM90, Bie89, Boy90, CG90, DS90, Eng90, GKSG91, Gro90, GNY90, GS90, KG91, Sne91, Syv91]. It has also helped identify subtle differences between similar protocols [BAN89].

On the other hand, a number of questions have been asked about the logic itself. What does "belief" mean? What is a "good key"? What kinds of properties of authentication protocols is the logic designed to study? And after a complete, formal proof of a protocol's correctness, no error has been discovered, but what exactly is it that has been proven? One reason for these questions is that the semantics given in [BAN89] is hard to relate to standard models of computation and belief. For example, in that model, part of a principal's state is an explicit set of the formulas it believes, and a principal is said to believe a formula if that formula is included in its set.

The main goal of this paper is to find a natural semantic model for the logic of authentication. We begin by trying to model the concepts the logic is trying to capture. For example, we give a possible-worlds definition of belief as a form of resource-bounded, defeasible knowledge (which is an interesting topic in itself). The process of constructing our model illuminates some of the intrinsic properties of these concepts. This insight enables us to reformulate the logic slightly without losing any of its essential features (protocols are analyzed with the reformulated logic in much the same way as they are with the original logic), and to give a sensible model of computation and semantics for this simplified logic.

We reformulate the logic in several ways. We remove some unnecessary mixing of semantic and implementation details in the original definitions and inference rules, clarifying the semantic properties required for the logic's soundness. We also introduce more direct definitions for several logical constructs, enabling us to dispense with an implicit assumption of honesty (that principals believe the messages they send are true), which is not well-defined in general. We substantially simplify the inference rules, so that all concepts are defined independently rather than jointly with other concepts. We also elaborate the set of inference rules, guided by our new semantics, and reformulate them as an axiomatization with modus ponens and necessitation as the only inference rules. The result is a simpler, more natural formulation of the logic with an easily understood semantics, and a potentially wider range of applications.

It should be noted that in this paper we are studying the relatively simple use of encryption in actual systems, and not the powerful tools of modern cryptography such as oblivious transfer and zero-knowledge proof systems. Until these tools become practical, it is important to have a tractable model for analysis of the protocols being implemented today. As in [BAN89], we make the simplifying assumption of perfect encryption, by which we mean that a principal must possess a key K in order to use K to encrypt or decrypt a message, or to obtain any information whatsoever about a message encrypted with K. The

intention is to study authentication protocols at a level of abstraction where the encryption technology being used does not matter. Many authentication protocols are subtle enough without considering the additional subtleties that arise when cryptography is introduced. Once authentication protocols are understood in the context of perfect encryption, an obvious next step is to relax this assumption and extend the logic and semantics to consider the effects of cryptographic insecurities on authentication protocols.

The rest of this paper is organized as follows. In Section 2, we review the logic of [BAN89]. We consider some of the possible improvements to the logic in Section 3, and reformulate it in Section 4. Our model appears in Section 5, and our semantics appears in Section 6. Finally, in Section 7 we continue a discussion of the definition of belief proposed in Section 6.

2 The Logic of Authentication

We now review the syntax and inference rules of the logic of authentication as it appears in [BAN89], although we use a more readable notation appearing in later presentations such as [BAN90]. Our reformulation of the logic in Sections 4-6 will simplify and clarify the logic's syntax and semantics.

2.1 Syntax and informal semantics

We begin with the language itself. The language assumes the existence of sets of constants to denote principals, encryption keys, and other things used in authentication protocols. The letters P, Q, R, and S are used to denote *principals*, the letter K is used to denote *encryption keys*, and the letters X and Y are used to denote *formulas* or *statements* in the language. The logic includes the following constructs:

P *believes* X: P believes X is true. X may be true or false, but P acts as though X is true.

P *sees* X: P has received X in a message. P can read the contents of X (possibly after decryption, assuming P has the needed keys), and can include X in messages to other principals.

P *said* X: P has sent X in a message. P believed that X was true when it sent the message.

P *controls* X: "P *has jurisdiction over* X." P is a trusted authority on the truth of X.[1]

$fresh(X)$: "X *is fresh.*" Time is divided into two *epochs*, the *past* and the *present*; the present begins with the start of the current execution of the current protocol. X is fresh if it is not contained in any message sent in the past. Verifying the freshness of messages guards against the replay of old messages.

$P \overset{K}{\leftrightarrow} Q$: "$K$ *is a shared key for* P *and* Q." K is a secure key for communication between P and Q, and it will never be discovered by any principal except P or Q, or a principal trusted by either P or Q.

$P \overset{X}{=} Q$: "X *is a shared secret between* P *and* Q." The expression X is a secret (such as a password) known only to P and Q, and possibly to principals trusted by them. P and Q can use X to prove their identities to one another.

$\{X\}_K$: "X *encrypted under* K." $\{X\}_K$ represents the message X encrypted using the key K. This is a shorthand for $\{X^P\}_K$, where P is a *from field* denoting the principal (usually clear from context) sending the message.[2] The mention of P is used only in implementing an assumption that each principal can recognize and ignore its own messages.

$\langle X \rangle_Y$: "X *combined with* Y." $\langle X \rangle_Y$ represents the message X combined in some way with Y, usually by concatenation. Y is intended to be a secret of some kind whose presence in the message proves the identity of the sender, just as the key used to encrypt a message can be used to prove the identity of the sender. The similarity of the roles played by keys and secrets in authentication motivates the similar notation.

In addition to these constructs, the conjunction and concatenation of two statements X and Y are denoted with a comma (X, Y). In the full paper, we also consider public key encryption as in [BAN89], but we have omitted it here since its treatment is similar to the treatment of shared keys.

2.2 Inference Rules

More information about the meaning of logical constructs can be deduced from a collection of inference rules in [BAN89]. As in [BAN89], we group these inference rules according to their function in the logic.

Message-meaning The message-meaning rules tell us how to deduce the identity of a message's sender from the encryption key or secret used: if $P \neq R$,

$$\frac{P \text{ believes } (Q \overset{K}{\leftrightarrow} P), \quad P \text{ sees } \{X^R\}_K}{P \text{ believes } (Q \text{ said } X)}$$

[1] We use the phrase "controls" here in order to remain consistent with notation used in later versions of [BAN89]. For the same reason, we denote "P believes X" by P *believes* X instead of $B_P X$ as many papers do.

[2] In [BAN89], the notation is $\{X\}_K$ *from* P.

$$\frac{P \text{ believes } (Q \overset{Y}{\rightleftharpoons} P),\ \ P \text{ sees } \langle X \rangle_Y}{P \text{ believes } (Q \text{ said } X)}$$

The first rule says that if P believes that P and Q share a key K and P sees a message encrypted with K (presumably known only to P and Q), then P believes that Q sent the message. However, P must be certain that it did not simply send the message to itself, and the side condition that $P \neq R$ reflects the assumption that a principal can detect and ignore its own messages. The second rule is a similar rule for shared secrets.

Nonce-verification The nonce-verification rule

$$\frac{P \text{ believes } (\mathit{fresh}(X)),\ \ P \text{ believes } (Q \text{ said } X)}{P \text{ believes } (Q \text{ believes } X)}$$

says that if P believes a message is fresh and that Q sent the message, then P believes that Q sent the message recently, and still believes the contents of the message. The name "nonce-verification" comes from the fact that the freshness of X is typically proven by including a nonce in the message X. A *nonce* is an expression (such as a timestamp) invented for the purpose of being fresh, and included in messages to prove their freshness.

Jurisdiction The jurisdiction rule

$$\frac{P \text{ believes } (Q \text{ controls } X),\ \ P \text{ believes } (Q \text{ believes } X)}{P \text{ believes } X}$$

captures what it means for Q to be a trusted authority on the truth of X.

Belief The belief rules

$$\frac{P \text{ believes } X,\ \ P \text{ believes } Y}{P \text{ believes } (X, Y)} \qquad \frac{P \text{ believes } (X, Y)}{P \text{ believes } X}$$

$$\frac{P \text{ believes } (Q \text{ believes } (X, Y))}{P \text{ believes } (Q \text{ believes } X)}$$

simply reflect an assumption that a principal believes a collection of statements iff it believes each of the statements individually.

Saying The saying rule

$$\frac{P \text{ believes } (Q \text{ said } (X, Y))}{P \text{ believes } (Q \text{ said } X)}$$

reflects the assumption that a principal is considered to have said each component of any message it has sent.

Seeing The seeing rules say that a principal sees all components of every message it sees, assuming it believes the necessary decryption key is a good key. These rules are

$$\frac{P \text{ sees } (X, Y)}{P \text{ sees } X} \qquad \frac{P \text{ sees } \langle X \rangle_Y}{P \text{ sees } X}$$

$$\frac{P \text{ believes } (Q \overset{K}{\leftrightarrow} P),\ \ P \text{ sees } \{X\}_K}{P \text{ sees } X}$$

In the original logic, a condition on from fields guarantees that principals recognize and ignore their own messages.

Freshness The freshness rule

$$\frac{P \text{ believes } (\mathit{fresh}(X))}{P \text{ believes } (\mathit{fresh}((X, Y)))}$$

says that any message with a fresh component is also fresh.

Shared keys and secrets Finally, the shared key rules

$$\frac{P \text{ believes } (R \overset{K}{\leftrightarrow} R')}{P \text{ believes } (R' \overset{K}{\leftrightarrow} R)}$$

$$\frac{P \text{ believes } (Q \text{ believes } (R \overset{K}{\leftrightarrow} R'))}{P \text{ believes } (Q \text{ believes } (R' \overset{K}{\leftrightarrow} R))}$$

and the shared secret rules

$$\frac{P \text{ believes } (R \overset{X}{\rightleftharpoons} R')}{P \text{ believes } (R' \overset{X}{\rightleftharpoons} R)}$$

$$\frac{P \text{ believes } (Q \text{ believes } (R \overset{X}{\rightleftharpoons} R'))}{P \text{ believes } (Q \text{ believes } (R' \overset{X}{\rightleftharpoons} R))}$$

say that shared keys and secrets are used in the same way in each direction.

2.3 Using the Logic

In [BAN89], protocols are written in a special idealized form. In most of the literature, an authentication protocol is described as a sequence of steps of the form "$P \rightarrow Q : X$" intended to denote the fact that P sends X to Q during this step (and that Q receives it). For example, the third step of the protocol in Figure 1 would be described by $A \rightarrow B : \{T_s, K_{ab}, A\}_{K_{bs}}$. The expression $\{T_s, K_{ab}, A\}_{K_{bs}}$ is intended to represent the actual bit string (the encrypted message) sent from A to B during the protocol's execution. It is only in the context of our understanding of the entire protocol that we know this bit string is to be interpreted as an encryption of the statement that K_{ab} is a good key for A and B.

The many possible interpretations of this bit string introduce an ambiguity that makes formal analysis of authentication protocols difficult. Consequently, instead of analyzing the informal descriptions of these protocols as they appear in the literature, we analyze protocols written in an idealized form. An *idealized protocol* in [BAN89] is a sequence of steps of the form "$P \rightarrow Q : X$" where X is an expression in the logical language. For example, the idealized version of the protocol illustrated in Figure 1 might be:

$$A \rightarrow S : A, B$$
$$S \rightarrow A : \{T_s, A \overset{K_{ab}}{\leftrightarrow} B, \{T_s, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}\}_{K_{as}}$$
$$A \rightarrow B : \{T_s, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$$

In fact, the first step would probably be omitted since it serves only to indicate the beginning of the protocol, and does not contribute anything of interest to the beliefs of the principals taking part in the protocol. Translating an actual protocol into an idealized protocol is usually quite easy, but the results of an analysis in this logic depend on the particular idealization of the protocol chosen.

To analyze a protocol, each step of the protocol is annotated with a formula in the logic. First, a formula called the *initial assumption* is written before the first statement. It is intended to describe the state of affairs at the beginning of an execution of the protocol. For example, in the case of Figure 1, the initial assumption might include

$$A \textit{ believes } (A \overset{K_{as}}{\leftrightarrow} S) \ \wedge \ B \textit{ believes } (B \overset{K_{bs}}{\leftrightarrow} S).$$

Next, a formula is written after each statement to describe the state of affairs after that step has been taken. These formulas are obtained in three ways:

1. Q *sees* X can be asserted after a step of the form $P \rightarrow Q : X$;
2. an assertion labeling one statement can be used to label any later statement; and
3. new assertions for a statement can be derived from old assertions via the inference rules.

The assertion annotating the final statement describes the outcome of the protocol. The goal of the analysis is typically to be able to derive an assertion for the last statement that expresses the protocol's expected correctness condition.

The soundness of this proof system—in particular, the soundness of step 2—depends on the fact that formulas in the logic are stable. A formula φ is said to be *stable* if it remains true once it becomes true. This means that, in every execution of the protocol, if φ is true after step k, then φ is true after every step $k' \geq k$. Stability is the justification for step 2 stating that formulas labeling one statement of a protocol can be used to label any later statement. The soundness of the nonce-verification rule stated in the preceding section also depends on stability, since once Q comes to believe X, it must continue to believe X. In our reformulation of the logic, formulas will no longer be guaranteed to be stable, and this will have a minor impact on the proof system just described (see Section 4.3).

3 Observations on the Logic

The main goal of this paper is to construct a simple semantic model for the logic in [BAN89]. In this section, we discuss some of the problems that can arise when constructing such a model, and we discuss how these problems can be resolved. In later sections, we will reformulate the logic along these lines, and provide a model and a semantics for this reformulated logic. Gong, Needham, and Yahalom [GNY90] have made some similar observations, but they were led to a more complicated logic, perhaps for lack of a suitable semantic basis. An early draft of Grove's work [Gro90] also suggests some similar ideas.

3.1 Implementation versus Semantics

Looking through the inference rules, we find confusion resulting from an unnecessary mixing of implementation and semantic issues.

An important instance of this mixing of issues concerns the role of secrecy in the definition of a good encryption key, and this point has created some controversy [Nes90, BAN90, Sne91, Syv91]. In [BAN89]—as in Section 2.1—a key K is defined to be a good shared key for P and Q if the only principals that discover K are P and Q and principals "trusted" by P or Q. This definition, however, is much stronger than is required for the soundness of the inference rules in Section 2.2. The only inference rule that really concerns good keys is the message-meaning rule. (We will consider the seeing rules in a moment.) It states that if P believes K is a good key for P and Q and if P receives a message encrypted with K, then P believes the message came from Q. The soundness of this rule does not depend on the secrecy of K, but on the property that P and Q are the only principals encrypting messages with K. Who discovers K is irrelevant: keeping K a secret is just one way of implementing this property. The formal semantics in [BAN89] defines a good key K in terms of who sends messages encrypted with K, and not in terms of keeping the key secret. In fact, this definition is

still too strong. In Section 6, we will define a good key K in terms of who actually uses K to encrypt messages. As long as P and Q are the only ones encrypting messages with K, other principals can send copies of these messages without violating the soundness of the message-meaning rule.

A second instance of the mixing of implementation and semantic issues concerns the message-meaning and the seeing rules. For example, the seeing rules say that if P believes K is a shared key for P and Q and P sees a message encrypted with K, then P sees the contents of this message. In this rule, it is implicit that if P believes K is a good key, then it actually possesses K and can use K to decrypt messages it receives. While some definitions of belief might have the property that "believing" a key is a good key implies "awareness" of the key, this is not true in general (for example, consider information-theoretic definitions of belief [FH88]). In Section 4.1, we will introduce a new construct P *has* K to denote the fact that P actually possesses the key K. In our model, P must hold K in order to use it to encrypt or decrypt messages (see axiom A8, Section 4.2). Now, possessing a key is a concept distinct from holding any beliefs about the quality of the key. This decoupling seems essential for obtaining a sound semantic basis. It also increases the power of the logic, as it becomes easy to analyze the Yahalom protocol [BAN89] and similar protocols.

3.2 Honesty

One of the assumptions made in [BAN89] is that all of the principals taking part in a protocol are *honest*, in the sense that each principal believes in the truth of each message it sends. This assumption is fundamental to the soundness of the nonce-verification rule, the rule stating that if P believes X is fresh and that Q has said X, then P believes Q believes X. We argue that honesty is not a reasonable assumption to make. Fortunately, we can remove the need for honesty by clarifying the meaning and intention of some of the logical constructs and inference rules.

Some reasonable protocols fail to satisfy the honesty assumption, such as those requiring a principal to forward a message it does not necessarily believe to be true. Since it might be appropriate to require that a principal believes in the truth of messages it isn't simply forwarding, in Section 4.1, we will introduce a new construct 'X', read "*forwarded* X," for distinguishing forwarded messages from newly constructed messages. We don't make this requirement ourselves in this paper, but the forwarding syntax seems to be convenient notation to introduce.

A more fundamental problem is that honesty is not a well-defined concept. Honesty is not just a syntactic or (thinking of protocols as state transition functions) functional restriction on protocols. Whether a protocol satisfies the honesty restriction depends on context; it may be an honest protocol in one environment but not in another. Fagin and Halpern [FH87] give a convincing definition of honesty they call *trusting**. They show that this definition can be viewed as the fixed point of a certain equation, but that a fixed point of this equation is not guaranteed to exist, and may not be unique even if it does. In this sense, they, too, conclude that honesty is not a well-defined notion. Even under conditions guaranteeing that honesty is well-defined, the difficulty of determining whether a protocol is in any sense honest seems to decrease the range of obviously legitimate applications of the logic. It might be quite appropriate to base an analysis in this logic on the initial assumption that principals believe other principals are honest, but it does not seem to be a good idea to accept honesty as a basic principle.

Ultimately, the best question to ask is where the logic depends on honesty. Honesty is a way of promoting saying to believing: if Q says X, then Q believes X. This promotion is used only in the nonce-verification and jurisdiction rules. As we now show, it is possible to do away with the honesty assumption altogether by clarifying the intent of these rules.

The nonce-verification rule concerns both honesty and freshness: if Q says X, then Q believes X; so if Q said X and X is fresh, then Q must have just said X and hence still believe X. We want to explain the concept of freshness in isolation. The logic has a construct P *said* X for stating that P sent X, but no construct for stating that P sent X in the present. In Section 4.1, we will introduce a new construct P *says* X to denote the fact that P has sent X in the present.[3] This construct will let us get to the heart of the meaning of freshness with a single axiom stating that if X is fresh and Q has said X, then Q has recently said X (see axiom A20, Section 4.2).

Returning to the honesty component of the nonce-verification rule, we notice that honesty is used only to promote Q's saying X to Q's believing X so that we can then apply the jurisdiction rule: roughly speaking, the nonce-verification rule says that if Q says X and X is fresh, then Q believes X; and the jurisdiction rule says that if Q believes X and Q is a trusted authority on X, then X must be true. In practice, however, if Q wants to convince another principal P that X is true, then Q must send a message to P saying that X is true (since Q can't convince P simply by believing X is true and saying

[3] The potential utility of a construct like P *says* X has been noted in [BAN89] in other contexts.

nothing). This intuition leads us to a much more direct axiomatization of jurisdiction: if Q has jurisdiction over X and Q has recently said X, then X is true (see axiom A15, Section 4.2). This more natural definition of jurisdiction obviates the need for honesty altogether, and potentially increases the practical applicability of the logic.[4]

3.3 Syntax and Inference Rules

In addition to changes to the logic motivated by the semantic issues discussed above, we improve the formulation of the syntax and inference rules in several other ways.

We start by giving a new syntax. In [BAN89], there is no distinction between the arbitrary expressions one can include in messages and formulas (expressions to which we can assign a truth value). For example, it is possible to prove that a principal believes a nonce, which doesn't make much sense. We make this distinction between arbitrary expressions and formulas.

We introduce all the propositional connectives (the original logic did not have negation, disjunction, or implication), making it possible to rewrite the inference rules of [BAN89] as axioms. This is aesthetically appealing, since it enables us to express the concepts the inference rules are meant to convey with formulas in the logic itself instead of using meta-linguistic tools like inference rules, and it also helps in comparing the logic to previous logics. We note that one reason negation was not allowed in [BAN89] was that the soundness of the nonce-verification rule depended on the stability of formulas (see Section 2.3). With the redefinition of jurisdiction, however, this is no longer a problem. On the other hand, the soundness of the annotation procedure does still depend on the stability of formulas used to annotate protocols, so we must use a restricted language when annotating protocols (see Section 4.3).

Finally, we state axioms that define the concepts in the logic independently and no longer in conjunction with each other, resulting in a noticeably simpler, more elegant proof system. In particular, most axioms no longer involve belief. Removing belief from the treatment of many concepts is an important simplification. (Independently, Grove [Gro90] and Gong, Needham, and Yahalom [GNY90] seem to have made a similar observation that the concept of belief is orthogonal to the other concepts in the logic.)

[4] There are also technical reasons for wanting to avoid the use of belief in the definition of jurisdiction. We discuss this in the full paper.

4 The Reformulated Logic

We now present our reformulation of the logic of authentication. Having already discussed the intuition behind the logical constructs and inference rules from [BAN89], as well as those we add ourselves, we will be brief.

4.1 Syntax

We begin with a definition of the language. Remember that in this paper we are analyzing idealized protocols in which the messages principals send to each other are not bit strings but are expressions in a particular language. We define a language $\mathcal{M}_{\mathcal{T}}$ of *messages* in which these idealized messages are written. Some of the statements in this language represent assertions such as $P \overset{K}{\leftrightarrow} Q$, and some of these statements simply represent data such as the constant 5 or the nonce T_s. Thus, some of the things that can be expressed in this language represent assertions to which it makes sense to assign a truth value, and some do not. We therefore identify a sublanguage $\mathcal{F}_{\mathcal{T}}$ of $\mathcal{M}_{\mathcal{T}}$ that we call the language of *formulas*, those expressions to which we can assign truth values. Since formulas are a sublanguage of messages, idealized protocols allow principals to send formulas to each other in messages.

We assume the existence of a set $\mathcal{T}$ of primitive terms. We assume that $\mathcal{T}$ contains several distinguished, disjoint sets of constant symbols: a set of *primitive propositions*, a set of *principals*, and a set of *shared keys*. The remaining constant symbols in $\mathcal{T}$ represent things like nonces.

We define messages and formulas in the logic by mutual induction. The language $\mathcal{M}_{\mathcal{T}}$ of messages is the smallest language over $\mathcal{T}$ satisfying the following conditions:

M1. φ is a message if φ is a formula,

M2. X is a message if X is a primitive term in $\mathcal{T}$,

M3. $(X_1, \ldots, X_k)$ is a message if $X_1, \ldots, X_k$ are messages,

M4. $\{X^P\}_K$ is a message if X is a message, K is a key, and P is a principal,

M5. $\langle X^P \rangle_Y$ is a message if X and Y are messages and P is a principal, and

M6. 'X' is a message if X is a message.

The language $\mathcal{F}_{\mathcal{T}}$ of formulas is the smallest language satisfying the following conditions:

F1. p is a formula if p is a primitive proposition,

F2. $\neg\varphi$ and $\varphi \wedge \varphi'$ are formulas if φ and φ' are formulas (other propositional connectives are defined in terms $\neg$ and $\wedge$),

F3. P *believes* φ and P *controls* φ are formulas if P is a principal and φ is a formula,

F4. P *sees* X, P *said* X, and P *says* X are formulas if P is a principal and X is a message,

F5. $P \overset{X}{\rightleftharpoons} Q$ is a formula if X is a message and P and Q are principals,

F6. $P \overset{K}{\leftrightarrow} Q$ is a formula if K is a key and P and Q are principals,

F7. $\mathit{fresh}(X)$ is a formula if X is a message, and

F8. P *has* K is a formula if P is a principal and K is a key.

It follows by condition M1 that the formulas $\mathcal{F}_\mathcal{T}$ form a sublanguage of the messages $\mathcal{M}_\mathcal{T}$. Several extensions to this language are possible (see Section 8).

4.2 Axiomatization

Our axiom system includes of two inference rules:

R1. Modus Ponens: From $\vdash \varphi$ and $\vdash \varphi \supset \psi$ infer $\vdash \psi$.

R2. Necessitation: From $\vdash \varphi$ infer $\vdash P$ *believes* φ.

The axioms are all the instances of tautologies of propositional calculus, and the following axiom schemas:

Belief For principals P and formulas φ and ψ,

A1. P *believes* $\varphi \wedge P$ *believes* $(\varphi \supset \psi) \supset P$ *believes* ψ

A2. P *believes* $\varphi \supset P$ *believes* $(P$ *believes* $\varphi)$

A3. $\neg P$ *believes* $\varphi \supset P$ *believes* $(\neg P$ *believes* $\varphi)$

The first axiom says that a principal believes all logical consequences of its beliefs, and the second and third axioms say that a principal can tell what it does and does not believe. Many properties follow from these axioms, including

A4. P *believes* $\varphi \wedge P$ *believes* $\varphi' \equiv P$ *believes* $(\varphi \wedge \varphi')$

On the other hand, $(P$ *believes* $\varphi) \supset \varphi$ does *not* hold in general: principals can be mistaken.

Message-meaning Keys and secrets are used to deduce the identity of the sender of a message: if $P \neq S$, then

A5. $P \overset{K}{\leftrightarrow} Q \wedge R$ *sees* $\{X^S\}_K \supset Q$ *said* X

A6. $P \overset{Y}{\rightleftharpoons} Q \wedge R$ *sees* $\langle X^S \rangle_Y \supset Q$ *said* X

Seeing A principal sees every component of every message it sees, providing it knows the necessary keys:

A7. P *sees* $(X_1, \ldots, X_k) \supset P$ *sees* X_i

A8. P *sees* $\{X^Q\}_K \wedge P$ *has* $K \supset P$ *sees* X

A9. P *sees* $\langle X^Q \rangle_S \supset P$ *sees* X

A10. P *sees* 'X' $\supset P$ *sees* X

A11. P *sees* $\{X^Q\}_K \wedge P$ *has* $K \supset P$ *believes* $(P$ *sees* $\{X^Q\}_K)$

The last axiom is a new axiom needed to reconstruct the message-meaning rule in [BAN89] from our axioms: in order for a principal P to believe it sees an encrypted message $\{X^Q\}_K$, it must hold the key K needed to decrypt $\{X^Q\}_K$, since otherwise, for all P knows, the encrypted message is really $\{Y^Q\}_K$ and not $\{X^Q\}_K$.

Saying If a principal has said (or recently said) a message, then the same is considered to be true of each component of that message:

A12. P *said* $(X_1, \ldots, X_k) \supset P$ *said* X_i

A13. P *said* $\langle X^Q \rangle_S \supset P$ *said* X

A14. P *said* 'X' $\wedge \neg P$ *sees* $X \supset P$ *said* X

Analogous axioms hold with P *says* X replacing P *said* X. The last axiom says that any principal misusing the forwarding syntax is held accountable for the contents of the "forwarded" message.

Jurisdiction This axiom reflects our redefinition of jurisdiction. It says that one can trust P's word on the truth of φ whenever P has jurisdiction over φ:

A15. P *controls* $\varphi \wedge P$ *says* $\varphi \supset \varphi$

Freshness A message is fresh if any component is fresh:

A16. $\mathit{fresh}(X_i) \supset \mathit{fresh}\,(X_1, \ldots, X_k)$

A17. $\mathit{fresh}(X) \supset \mathit{fresh}\,(\{X^Q\}_K)$

A18. $\mathit{fresh}(X) \supset \mathit{fresh}\,(\langle X^Q \rangle_S)$

A19. $\mathit{fresh}(X) \supset \mathit{fresh}\,(\text{'}X\text{'})$

Nonce-verification The nonce-verification axiom is now essentially a definition of freshness. A fresh message must have been recently said:

A20. $fresh(X) \wedge P\ said\ X \supset P\ says\ X$

Shared keys and secrets Keys and secrets shared between two principals can be used in either direction:

A21. $R \overset{K}{\leftrightarrow} R' \equiv R' \overset{K}{\leftrightarrow} R$

A22. $R \overset{K}{=\!=} R' \equiv R' \overset{K}{=\!=} R$

When compared to the shared key and secret rules in Section 2.2, this axiom is a nice example of how much more compactly our reformulation expresses the same concepts as the original logic.

4.3 Using the Logic

The use of this proof system to analyze protocols is essentially identical to the use of the original proof system described in Section 2.3, but there are two novelties.

Since we have added negation and primitive propositions to the language, it is now possible to write unstable formulas, which may hold after one protocol step and not after a later step. Now we must require that the formulas annotating protocols are stable. In fact, since the formulas Q *sees* X asserted after a step of the form $P \rightarrow Q : X$ are always stable, the stability of formulas annotating protocols is only an issue when stating the initial assumptions. Although stability is a semantic concept, simple linguistic restrictions (such as avoiding the use of the belief operator in the scope of negation) usually suffice in practice, and are no more restrictive than the original language in [BAN89].

Since we have introduced the notion of key possession and acquiring a key, we need to provide a mechanism for proving that a principal has a key. We extend the syntax of idealized protocols to include steps of the form "$P : newkey(K)$," denoting the fact that P has added the key K to its key set, in addition to steps of the form $P \rightarrow Q : X$. We also extend the protocol annotation procedure to allow P *has* K to be asserted after a step of the form $P : newkey(K)$.

5 Model of Computation

We briefly sketch our model of computation. A system consists of a finite collection of *system principals* $P_1, \ldots, P_n$ who communicate by sending messages to each other. We also assume the existence of a distinguished principal P_e, called the *environment*, that, for example, represents other principals trying to attack an authentication protocol.

At any given time, a principal is in some *local state*; we associate a set of local states with each principal. A *global state* is a tuple $(s_e, s_1, \ldots, s_n)$ of local states, one s_e for the environment P_e and one s_i for each P_i. We assume that the environment state encodes all interesting aspects of the global state that cannot be deduced from the local states of the system principals (such as the messages in transit between principals).

In any given state, any principal can change its local state—and perhaps the environment state—by performing an *action*. We associate with each principal a set of actions that principal can perform. An action is identified with a state-transition relation (in which only the principal's and environment's states are changed). A *local protocol* for P is a function from P's local state to the next action P is to perform. A *protocol* is a tuple $(A_e, A_i, \ldots, A_n)$ of local protocols, one A_e for P_e and one A_i for each P_i.

A *run* is an infinite sequence of global states. A *system* is a set $\mathcal{R}$ of runs, typically the set of executions of a given protocol. Integer times are assigned to each state in a run: the first state of run r is assigned some time $k_r \leq 0$, and the kth state is assigned time $k_r + (k-1)$. We consider the state at time 0 to be the first state of the current epoch (the first state of the current authentication), and we call it the *initial state*. We denote the global state at time k in run r by $r(k)$, and the local state of P_i in $r(k)$ by $r_i(k)$. We refer to the ordered pair (r, k) consisting of a run r and a time k as a *point*.

We assume a principal P_i's local state includes a *local history* (the sequence of all actions the principal has ever performed) and a *key set* (the set of keys the principal holds; we will return to this set in a moment). The environment P_e's state includes a *global history* (the sequence of actions any principal has performed), a *key set*, and a *message buffer* m_i for each system principal P_i containing all messages sent to P_i but not yet delivered. We assume that principals' histories and message buffers are empty in the first state of a run (which might be different from the initial state), but the values of other components depend on the application being modeled.

We assume that the set of actions a principal P can perform includes the following actions:

1. $send(m, Q)$: This denotes P's sending of the message m to Q. The message m is added to Q's message buffer.

2. $receive()$: This denotes P's receipt of a message. Some message m is nondeterministically chosen and deleted from P's message buffer. (We don't consider $receive()$ to be an environment action.)

3. *newkey*(K): This denotes P's coming into possession of a new key. The key K is added to P's key set.

Furthermore, each action appends itself to the end of the principal's local history and environment's global history. We actually append *receive*(m) to these histories, in order to tag the *receive*() action with the message m returned. Similarly, in the global history, we tag actions with the name of the principal performing the action. It is convenient to define the *messages P has received* by time k in a run r to be the set of messages m such that *receive*(m) appears in P's local history in $r(k)$, and to define the set of *messages P has sent* similarly.

In this paper, we make the simplifying assumption of *perfect encryption*: a principal must have K in its key set in order to use K to encrypt or decrypt a message, or to obtain any information at all about a message encrypted with K. Since cryptographic security statements are often quite difficult, and since many authentication protocols are subtle enough without worrying about these additional difficulties, it seems reasonable to assume perfect encryption operators for the time being, and to consider the effect of cryptographic weaknesses on authentication protocols after the protocols themselves are understood.

Key sets together with the assumption of perfect encryption give us a syntactic way to determine from a principal's local state what messages the principal can encrypt and decrypt. While a principal may attack an encrypted message by generating many different keys and trying to decrypt the message with each key, the principal can only generate a small fraction of the possible keys with a polynomial amount of computation. The key set is used to determine what fraction of the key space the principal has discovered: we require that when a principal uses a key to encrypt or decrypt a message, that key must appear in the principal's key set. Each time the principal generates a new key K to use to attack an encrypted message, it first adds the key to its key set using the *newkey*(K) action, and then attempts to decrypt the message using K and other keys in its key set.

This intended use of key sets leads us to define an operation that takes a message M and a principal P's key set $\mathcal{K}$ and returns the components (the submessages) of M that P can read. These components include contents of any encrypted component $\{X^Q\}_K$ for which P has the key K needed to decrypt the message. We define *seen-submsgs*$_{\mathcal{K}}(M)$ to be the union of the singleton set $\{M\}$ and

1. *seen-submsgs*$_{\mathcal{K}}(X_1) \cup \cdots \cup$ *seen-submsgs*$_{\mathcal{K}}(X_k)$ if $M = (X_1, \cdots, X_k)$,
2. *seen-submsgs*$_{\mathcal{K}}(X)$ if $M = \{X^Q\}_K$ and $K \in \mathcal{K}$,
3. *seen-submsgs*$_{\mathcal{K}}(X)$ if $M = \langle X^Q \rangle_S$, and
4. *seen-submsgs*$_{\mathcal{K}}(X)$ if $M =$ 'X'.

We extend this operation from a single message to a set of messages in the obvious way.

Similarly, we can define an operation that takes a message M, a principal P's key set $\mathcal{K}$, and the set $\mathcal{M}$ of all messages received by P so far, and returns the components of M we consider P to have said as the result of sending M. Our definition has the property that if P has said $\{M\}_K$ and P holds the key K used to encrypt the message, then P is considered to have said M as well. Of course, if P is simply forwarding $\{M\}_K$ for some other principal, then it seems a bit harsh to force P to accept responsibility for saying M, but P can always absolve itself of this responsibility by using the forwarding syntax and sending '$\{M\}_K$' instead of $\{M\}_K$. We define *said-submsgs*$_{\mathcal{K},\mathcal{M}}(M)$ to be the union of the singleton set $\{M\}$ and

1. *said-submsgs*$_{\mathcal{K},\mathcal{M}}(X_1) \cup \cdots \cup$ *said-submsgs*$_{\mathcal{K},\mathcal{M}}(X_k)$ if $M = (X_1, \cdots, X_k)$,
2. *said-submsgs*$_{\mathcal{K},\mathcal{M}}(X)$ if $M = \{X^Q\}_K$ and $K \in \mathcal{K}$,
3. *said-submsgs*$_{\mathcal{K},\mathcal{M}}(X)$ if $M = \langle X^Q \rangle_S$, and
4. *said-submsgs*$_{\mathcal{K},\mathcal{M}}(X)$ if $M =$ 'X' and $X \notin$ *seen-submsgs*$_{\mathcal{K}}(\mathcal{M})$.

Part 4 says that a principal (such as a malicious environment) misusing the forwarding notation is held to account for the message being "forwarded."

These definitions now enable us to state several syntactic restrictions on runs. Some of these properties follow from the model described above, but we state them explicitly since the soundness of our semantics depends on them. Given any run r and any time k, if $\mathcal{K}$ is P's key set at time k and $\mathcal{M}$ is the set of messages P has received before time k, then we require that

1. A principal's key set never decreases: If $\mathcal{K}'$ is P's key set at time $k' \leq k$, then $\mathcal{K}' \subseteq \mathcal{K}$.
2. A message must be sent before it is received: If *receive*(M) appears in P's local history at time k, then *send*(M, P) appears in some principal Q's local history at time k.
3. A principal must possess keys it uses for encryption: Suppose that the action *send*(M, Q) appears in P's local history at time k and that $\{X^R\}_K \in$ *said-submsgs*$_{\mathcal{K},\mathcal{M}}(M)$. Then either $\{X^R\}_K \in$ *seen-submsgs*$_{\mathcal{K}}(\mathcal{M})$ or $K \in \mathcal{K}$.
4. A system principal sets from fields correctly: if *send*(M, Q) appears in P's local history at

time k and $\{X^R\}_K \in \mathit{said\text{-}submsgs}_{\mathcal{K},\mathcal{M}}(M)$, then $P = R$ or $\{X^R\}_K \mathit{seen\text{-}submsgs}_{\mathcal{K}}(\mathcal{M})$, and similarly for $\langle X^R \rangle_Y$.

5. A system principal must see messages it forwards: if $\mathit{send}(M, Q)$ appears in P's local history at time k and 'X' $\in \mathit{said\text{-}submsgs}_{\mathcal{K},\mathcal{M}}(M)$, then $X \in \mathit{seen\text{-}submsgs}_{\mathcal{K}}(\mathcal{M})$.

Condition 3 says that if P says $\{M\}_K$, then either P has received $\{M\}_K$ in a previous message, or P has the key K needed to construct $\{M\}_K$ (and hence is also considered to have said the contents M of $\{M\}_K$). A system principal P's behavior is restricted more than the environment's: condition 4 says that P does not misuse the from field in a message, and condition 5 says that P forwards only what it has previously received.

6 Semantics

We now give our semantics for the logic. Since we have already discussed most of the issues involved, we will be brief. Fix a system $\mathcal{R}$, and fix an *interpretation* π of the primitive propositions that maps each $p \in \Phi$ to the set of points $\pi(p)$ in $\mathcal{R}$ at which p is true. We define the truth of φ at (r, k), denoted $(r, k) \models \varphi$, by induction on the structure of φ. First, we define

$(r, k) \models p$ iff $(r, k) \in \pi(p)$ for primitive $p \in \Phi$,

$(r, k) \models \varphi \wedge \varphi'$ iff $(r, k) \models \varphi$ and $(r, k) \models \varphi'$, and

$(r, k) \models \neg\varphi$ iff $(r, k) \not\models \varphi$.

For the remaining constructs, we proceed as follows.

Seeing A principal sees all components it can read in the messages it has received. We define P *sees* X at (r, k) by

$$(r, k) \models P \mathit{\ sees\ } X$$

iff, for some message M, at time k in r

1. $\mathit{receive}(M)$ appears in P's local history, and

2. $X \in \mathit{seen\text{-}submsgs}_{\mathcal{K}}(M)$, where $\mathcal{K}$ is P's key set.

As P comes into possession of more keys, it is able to decrypt more of the messages it has received, and see more of their contents.

Saying In the process of constructing a message, a principal may construct many message components, and the principal is considered to have said each of these components. We define P *has said* X at (r, k) by

$$(r, k) \models P \mathit{\ said\ } X$$

iff, for some message M, at some time $k' \leq k$ in r

1. P performs $\mathit{send}(M, Q)$, and

2. $X \in \mathit{said\text{-}submsgs}_{\mathcal{K},\mathcal{M}}(M)$, where $\mathcal{K}$ is P's key set and $\mathcal{M}$ is the set of messages P has received.

If P sends $\{X^Q\}_K$, then P says X only if it possessed K when it sent $\{X^Q\}_K$, and does not come to say X at a later point simply as a consequence of having acquired K at that point.

We define P *has recently said* X in (r, k), denoted

$$(r, k) \models P \mathit{\ says\ } X,$$

in the same way, except that k' is restricted to the range $0 \leq k' \leq k$. In other words, the message M containing X must have been sent in the current epoch.

Jurisdiction A principal P is an authority on φ if φ is true whenever P says φ is true. We define P *has jurisdiction over* φ at (r, k) by

$$(r, k) \models P \mathit{\ controls\ } \varphi$$

iff $(r, k') \models P \mathit{\ says\ } \varphi$ implies $(r, k') \models \varphi$ for all $k' \geq 0$. Notice that P has jurisdiction over φ at one point of a run iff it does at all points (in the current epoch) of the run. For this reason, P *controls* φ is more than a shorthand for P *says* $\varphi \supset \varphi$.

Freshness A message is fresh if it has not been sent in the past, and has not been contained in a message sent in the past. Given a set $\mathcal{M}$ of messages, it is convenient to define $\mathit{submsgs}(\mathcal{M})$ to be the set of all submessages of the messages in $\mathcal{M}$. (The full paper contains a formal definition by induction.) Let $\mathcal{M}(r, 0)$ be the set of messages sent by any principal by time 0 in r. We define X *is fresh* in (r, k) by

$$(r, k) \models \mathit{fresh}(X)$$

iff $X \notin \mathit{submsgs}(\mathcal{M}(r, 0))$.

Shared keys and secrets A key K is a shared key for P and Q if P and Q are the only principals encrypting messages with K. Other principals sending messages encrypted with K must have received them from other principals. We define K *is a shared key for* P *and* Q at (r, k) by

$$(r, k) \models P \overset{K}{\leftrightarrow} Q$$

iff, for all k', $(r,k') \models R$ *said* $\{X^S\}_K$ implies either $(r,k') \models R$ *sees* $\{X^S\}_K$ or $R \in \{P,Q\}$. Notice that the universal quantification over all k', and not just $k' \geq 0$, means that a good key for one pair of principals in one epoch cannot be a good key for another pair in another epoch.

Similarly, we define X *is a shared secret between* P *and* Q at (r,k) by

$$(r,k) \models P \overset{X}{=} Q$$

iff, for all k', $(r,k') \models R$ *said* $\langle Y^S \rangle_X$ implies either $(r,k') \models R$ *sees* $\langle Y^S \rangle_X$ or $R \in \{P,Q\}$.

Belief We now turn our attention to the definition of belief, the most interesting of our definitions. The classical definition of belief is in terms of *possible-worlds* and dates back to Hintikka [Hin62], but recently it has been formulated in the context of distributed systems [CM86, HM90]. The intuition is that in any given world (or point), a principal P considers a number of other worlds to be *possible*, and that P *believes* a fact φ if φ is true in all worlds P considers possible. For example, suppose that at any given point, P considers another point possible if the two points are *indistinguishable* to P, meaning that P has the same local state at both points. This definition of possible worlds yields the standard possible-worlds definition of *knowledge*. According to this definition of knowledge, P knows φ iff the truth of φ follows from the information recorded in its local state, since φ must be true at all points in which P has this local state. Consequently, this definition satisfies the axiom (P *believes* φ) $\supset \varphi$: if P believes φ at (r,k), then φ is true at all points indistinguishable from (r,k) by P, including the point (r,k) itself.

This definition of knowledge, however, does not seem an appropriate definition for belief in the context of authentication protocols. For example, given the model of computation we have defined, it can never be the case that a desirable initial assumption of the form P *believes* $P \overset{K}{\leftrightarrow} Q$ is true. To see this, notice that at any point (r,k) there is a point (r',k) indistinguishable to P in which a malicious environment randomly generating keys has stumbled upon the key K by dumb luck and started using K to encrypt messages. It follows that P cannot believe K is a good key for P and Q at (r,k) since there is a point P considers possible at (r,k) at which K is *not* a good key.

Our intuition is that a principal with preconceived beliefs like $P \overset{K}{\leftrightarrow} Q$ is restricting its set of possible worlds to those in which its preconceptions are true, and not just to those it considers indistinguishable. We view P_i's initial beliefs as identifying a set G_i of "good" runs, those in which these initial beliefs are initially true. We will define the set of points that P_i considers possible at a point to be the indistinguishable points of runs in G_i, and this will determine a notion of *belief relative to* $\mathcal{G} = (G_1, \ldots, G_n)$. We leave the problem of actually calculating $\mathcal{G}$ for the next section.

First, however, a further refinement is required to capture our assumption of perfect cryptography. Specifically, before computing the points a principal considers possible, we must hide the contents of unreadable encrypted messages. To see why, suppose P's local state contains $\{X^Q\}_K$ but that P does not possess K and hence cannot read X. If we do not hide unreadable encrypted messages, then P's local state will contain $\{X^Q\}_K$ at all points it considers possible, and hence P will believe that $\{X^Q\}_K$ contains X even though P cannot read X! We therefore define an operation $hide(s)$ that takes a principal's local state s and hides all messages in s that it cannot read. (The full paper contains a formal definition by induction.) For example, suppose P's local state s contains a message $(\{X^Q\}_K, \{Y^R\}_L)$, and that P's key set does not contain K. In the state $hide(s)$, this message is replaced by something like $(\perp, \{Y^R\}_L)$, where $\perp$ is intended to represent encrypted messages that cannot be read by a principal with the keys in its possession.

We now define a principal P_i's beliefs (relative to $\mathcal{G} = (G_1, \ldots, G_n)$). Define the possibility relation $\sim_i$ for P_i by

$$(r,k) \sim_i (r',k')$$

iff $r' \in G_i$ is a good run and $hide(r_i(k)) = hide(r'_i(k'))$. This says that the points a principal considers possible are those points of good runs that the principal considers indistinguishable from the current point after hiding the encrypted messages. Finally, we say that P_i *believes* φ *(relative to* $\mathcal{G}$*)* at a point (r,k), denoted

$$(r,k) \models P_i \textit{ believes } \varphi,$$

iff $(r',k') \models \varphi$ for all (r',k') such that $(r,k) \sim_i (r',k')$.

Soundness of the axioms We can now prove that the axiomatization given in Section 4.2 is sound relative to the model and semantics given in Sections 5 and 6:

Theorem 1: The axiomatization is sound.

One might also ask whether the axiomatization is complete. We believe the answer is "no." For example,

$$P \textit{ controls } (P \textit{ has } K) \ \wedge \ P \textit{ says } (P \textit{ has } K, \{X^P\}_K) \supset P \textit{ says } X.$$

is a valid formula but it does not seem to be derivable. This possible incompleteness, however, does not concern us here. Our goal has been to find a model to explain the logic of authentication, and the fact that the axiomatization is sound leads us to believe that we have done so.

7 Choosing the Good Runs

We now consider how to construct the sets of good runs used in Section 6 to define belief. The proof of soundness in Theorem 1 does not depend on any particular choice of sets; they are simply a mechanism to guarantee that axioms A2 and A3 are sound.

On the other hand, our original intuition was that if a principal P_i has a preconceived belief that φ is true, then it is effectively restricting its set of possible worlds to to those in which φ is true at time 0. To make this precise, let us fix a system $\mathcal{R}$, and let us fix for each principal P_i a set I_i of initial assumptions, where I_i is a set of formulas of the form P_i *believes* φ, and let $\mathcal{I} = (I_1, \ldots, I_n)$. Notice that if φ holds at all time 0 points in G_i, then P_i *believes* φ holds at all time 0 points of $\mathcal{R}$. Thus, given a vector $\mathcal{G} = (G_1, \ldots, G_n)$, we say that $\mathcal{G}$ *supports* $\mathcal{I}$ if the formulas in I_i hold at all time 0 points of $\mathcal{R}$ relative to $\mathcal{G}$.

We give a construction that is guaranteed to yield a vector $\mathcal{G}$ supporting $\mathcal{I}$, but in order for our construction to be well-defined, we must make one restriction on the language of initial assumptions:

I1. No formula P_i *believes* φ appears within the scope of a negation symbol.

This means that we do not allow an initial assumption like "P_i does *not* believe K is a good key" (although we do allow "P_i believes K is not a good key" if it is of any interest). This is, of course, a restriction, but it does not appear to be significant in practice. In every application of this logic that we are aware of, the initial assumptions satisfy this restriction (and so do the proof's conclusions, for that matter).

Our construction yields a vector $\mathcal{G}$ supporting $\mathcal{I}$, but there are potentially many such vectors, so how are we to choose among them? Given two vectors $\mathcal{G}' = (G'_1, \ldots, G'_n)$ and $\mathcal{G} = (G_1, \ldots, G_n)$, we can order them by set inclusion:

$$\mathcal{G}' \leq \mathcal{G} \text{ iff } G'_i \subseteq G_i \text{ for all } i.$$

Given any formula of the form P_i *believes* φ satisfying I1, it is easy to see that if P_i believes φ relative to $\mathcal{G}$, then P_i believes φ relative to every $\mathcal{G}' \leq \mathcal{G}$: since $G'_i \subseteq G_i$, every point P_i considers possible relative to $\mathcal{G}'$ is a point it considers possible relative to $\mathcal{G}$, so if P_i believes φ relative to $\mathcal{G}$, then P_i believes φ relative to $\mathcal{G}'$. A vector $\mathcal{G}$ supporting $\mathcal{I}$ is *optimum* if it is the maximum of all vectors supporting $\mathcal{I}$. Given an optimum $\mathcal{G}$, if P_i believes φ relative to $\mathcal{G}$, then P_i believes φ relative to any vector supporting $\mathcal{I}$. Thus, relative to $\mathcal{G}$, P_i initially believes only its initial beliefs and all beliefs that necessarily follow from them. A good construction should yield an optimum $\mathcal{G}$, assuming an optimum $\mathcal{G}$ exists.

While our construction depends on I1, there is convincing evidence that if $\mathcal{I}$ violates I1, then there is no definition of belief supporting $\mathcal{I}$ that is best in any convincing sense, regardless of whether we try to use our technique of defining belief in terms of sets of good runs or not. Halpern and Moses [HM84] consider the problem of characterizing the state of knowledge (not belief) of an agent who "knows only α." The idea is that if α is a formula describing all of the information explicitly available to the agent—for example, α might describe a principal's initial beliefs about some, but not all, encryption keys—then there should be a unique state of knowledge that characterizes everything this agent knows. In particular, for every question of the form "Does the agent know φ?" this state should determine a unique yes or no answer. Halpern and Moses give several convincing, equivalent characterizations of this desired unique state of knowledge, and one of them is a maximum model condition similar to our definition of optimality. They effectively show that if negation is included in the logic, then—even in a system with a *single* principal and formulas with a *single* level of belief—this unique state of knowledge does not exist for all α. One α they give as an example is the formula that "P knows φ or P knows φ'." There is one state of knowledge in which P knows φ and not φ', and a second state of knowledge in which P knows φ' and not φ, but neither state is obviously superior to the other. Since we have defined disjunction (and implication) in terms of negation, we avoid negation in order to avoid troublesome α's like the one given above.

We now define an iterative construction of the vector $\mathcal{G}$. Given the restriction I1 and belief axioms A2 and A3, we can assume without loss of generality that every formula in I_i is of the form P_i *believes* $\cdots P_k$ *believes* φ, where φ is a formula not involving belief. Let I_i^j be the set of formulas P_i *believes* $\cdots P_k$ *believes* φ in I_i with j levels of belief. Since we can always add a formula like P_i *believes* $\cdots P_i$ *believes true* to I_i^j, we can assume the I_i^j are nonempty. For each j, define $\mathcal{G}^j = (G_1^j, \ldots, G_n^j)$ by $G_i^0 = \mathcal{R}$ and

$$G_i^j = G_i^{j-1} \cap \{r : (r,0) \models \varphi \text{ relative to } \mathcal{G}^{j-1}, \text{ for every formula } P_i \textit{ believes } \varphi \text{ in } I_i^j\}$$

Finally, define $\hat{\mathcal{G}} = (\hat{G}_1, \ldots, \hat{G}_n)$ by $\hat{G}_i = \cap_j G_i^j$. We can prove that:

Theorem 2: If $\mathcal{I}$ satisfies I1, then $\hat{\mathcal{G}}$ supports $\mathcal{I}$.

This shows that our construction yields a model of belief in which the initial assumptions are satisfied, but is it optimum?

If the initial assumptions in $\mathcal{I}$ do not involve nested belief, then it is not too difficult to show that our construction yields the optimum $\mathcal{G}$ supporting $\mathcal{I}$. This condition is satisfied by all of the example protocols considered so far. In fact, in this case, our definition of belief is essentially equivalent to a definition of belief as defeasible knowledge proposed by Shoham and Moses [SM89] (special cases of their definition appear in [MT88]). They suggest defining P_i believes φ given the assumption α by

$$B_i(\varphi, \alpha) = K_i(\alpha \supset \varphi)$$

where $B_i\psi$ and $K_i\psi$ denote P_i's belief and knowledge of ψ, respectively. In other words, the agent knows that either φ is true, or something unusual has occurred (α is false). Our definition of P_i *believes* φ corresponds to their definition $B_i(\varphi, \alpha)$ when α is the condition that I_i holds at time 0.

Shoham and Moses note that it is possible to derive $K_i\neg\alpha \supset B_i(\varphi, \alpha)$; that is, P_i believes φ whenever P_i knows its assumptions are violated, even if it knows φ is false (which is rather strange). This motivates them to consider the definition

$$B_i(\varphi, \alpha) = K_i(\alpha \supset \varphi) \wedge (K_i\neg\alpha \supset K_i\varphi).$$

It says that if P_i knows its assumptions are violated, then it believes φ only if it knows φ is true. In the full paper, we show how to modify our definition of belief slightly to yield a definition that coincides with this definition as well.

Our formulation of belief has one significant advantage over the definitions given by Shoham and Moses. Their formulation necessarily requires that the assumption α is a propositional formula (at least one not involving belief): their definition has an unresolved circularity if α concerns belief. On the other hand, our formulation allows the initial assumption I_i to contain arbitrary nesting of belief formulas.

The obvious question now is how well our construction does when we *do* allow nested belief. Actually, even if we allow just two levels of belief, there is in general no optimum choice of good runs supporting $\mathcal{I}$, and hence no construction can do well in general. The example we give in detail in the full paper involves a coin-tossing situation with three principals P_1, P_2, and P_3. The state of each principal consists of the outcome of a single coin toss, but P_1 and P_3 disagree about the outcome of P_2's coin toss. Principal P_1 believes the coin landed tails and believes P_3 believes the same thing, while P_3 believes the coin landed heads and believes P_1 believes so, too. We show that either the set G_1 can contain the run in which the coin landed tails, or the set G_3 can contain the run in which the coin landed heads, but not both. Consequently, there can be no maximum $\mathcal{G}$ supporting these initial assumptions.

This example, however, is rather strange, since it *requires* that P_1 and P_3 be mistaken about each other's beliefs. The whole purpose of stating initial conditions is to restrict the initial state of affairs in some uniform way that guarantees that a protocol works correctly, and not to model errors. This motivates consideration of the following condition:

I2. If I_i contains P_i *believes* (P_j *believes* φ), then I_j contains P_j *believes* φ.

This basically says that the initial assumptions of one principal do not contain errors about the beliefs of the others. While we know of no proof with initial assumptions involving nested beliefs, the properties I1 and I2 are satisfied by the conclusions of proofs that yield formulas involving nested beliefs as conclusions. This suggests that they should not stand in the way of the possibility of composing protocols sequentially, and proving the correctness of the composition by composing the proofs of the individual protocols. We can prove:

Theorem 3: If $\mathcal{I}$ satisfies I1 and I2, then $\hat{\mathcal{G}}$ supports $\mathcal{I}$ and $\hat{\mathcal{G}}$ is optimum.

We have shown that when I1 holds, our construction yields a vector $\mathcal{G}$ supporting $\mathcal{I}$. We have also shown that there is in general no optimum $\mathcal{G}$ supporting $\mathcal{I}$ when I2 does not hold, and that our construction yields the optimum $\mathcal{G}$ when it does hold. In this sense, our construction seems to do as well as can be expected.

8 Extensions

Several useful extensions of the logic are possible. We sketch two of them here briefly.

So far, we have brushed over the fact that an idealized protocol is written schematically, and that some of the symbols in its description have different values in different runs. For example, in the idealization $A \rightarrow B : \{T_s, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$ of the third step of the protocol in Figure 1, the symbol K_{ab} is intended to be

the key generated by the server S in a particular run, and not the same key in every run. Consequently, it is convenient to add to the logic a distinguished set of symbols we call *parameters*, and to allow these parameters to appear in formulas and idealized protocols. We assume that a run uniquely determines the value of each parameter in the run. To compute the truth of a formula φ at a point (r, k), we first replace the parameters in φ with their values in the run r, and then follow the inductive definition given in Section 6. For example, defining the truth of a formula like A *believes* $A \overset{K_{ab}}{\leftrightarrow} B$ at a point (r, k) proceeds as follows: if K is the value of the parameter K_{ab} in r, then A *believes* $A \overset{K_{ab}}{\leftrightarrow} B$ holds at (r, k) if $A \overset{K}{\leftrightarrow} B$ holds at all points A considers possible at (r, k). Of course, the parameter K_{ab} might actually assume different values at these points, but this will not happen in the common case when its value is determined by A's local state at (r, k).

It is also convenient to allow universal quantification over constants (such as keys, nonces, or principals), so that we can write formulas such as

$$A \textit{ believes } \forall K.(S \textit{ controls } A \overset{K}{\leftrightarrow} B)$$

to express the fact that A trusts the server S to generate good encryption keys for communication between A and B. Since the set of all keys is typically finite in practice, this is equivalent to a finite conjunction of formulas already in our language, but quantification allows much more compact formulas. There have even been uses of quantification over formulas [ABKL90], but in practice a general assertion of the form

$$A \textit{ believes } \forall \varphi.(S \textit{ controls } \varphi)$$

can often be replaced with a more specific assertion such as the one above. Adding universal quantification over constants is straightforward and requires minimal changes to our work.

9 Conclusion

We have constructed a model and a semantics for a slightly reformulated version of the logic of authentication in [BAN89] that we believe captures and clarifies much of the original authors' intuition. Interesting problems to consider for the future include elaborating the logic and semantics to deal with secrecy (in addition to authentication), and relaxing the assumption of perfect encryption. As another possibility, our bibliography contains references to several logics for authentication and security such as [GNY90], and it would be interesting to know whether slightly modified versions of our model could be used to explain these logics and how they differ from the logic in [BAN89].

Acknowledgements: We thank Mike Burrows, Murray Mazer, Yoram Moses, Craig Schaffert, and Margaret Tuttle for stimulating discussions and constructive comments.

References

[ABKL90] Martín Abadi, Michael Burrows, Charles Kaufman, and Butler Lampson. Authentication and delegation with smart-cards. Research Report 67, DEC Systems Research Center, October 1990.

[AM90] C. l'Anson and C. Mitchell. Security defects in CCITT recommendation X.509—the directory authentication framework. *Computer Communication Review*, 20(2), 1990.

[BAN89] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *Proceedings of the Royal Society of London A*, 426:233–271, 1989. A preliminary version appeared as Research Report 39, DEC Systems Research Center, Palo Alto, February 1989.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. Rejoinder to Nessett. *Operating Systems Review*, 24(2), 1990.

[Bie89] P. Bieber. *Aspect Epistémiques des Protocoles Cryptographiques.* PhD thesis, Université Paul-Sabatier de Toulouse, 1989.

[Boy90] C. Boyd. Towards a formal framework for authentication. Manuscript, University of Manchester, 1990.

[CG90] P.-C. Cheng and V. D. Gligor. On the formal specification and verification of a multiparty session protocol. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, CA, 1990.

[CM86] K. Mani Chandy and Jayadev Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.

[DS90] D. Davis and R. Swick. Kerberos authentication and workstation services and Kerberos authentication at Project Athena. Technical Memorandum 424, MIT Laboratory for Computer Science, 1990.

[Eng90] U. Engberg. Analyzing authentication protocols. Technical Report DAIMI IR-97, Aarhus University, 1990.

[FH87] Ronald Fagin and Joseph Y. Halpern. I'm OK if you're OK: On the notion of trusting communication. In *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science*, pages 280–292, 1987. Also available as IBM Research Report RJ 5600.

[FH88] Ronald Fagin and Joseph Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988. Also available as IBM Research Report RJ 4657.

[GKSG91] V. D. Gligor, R. Kailar, S. Stubblebine, and L. Gong. Logics for cryptographic protocols—virtures and limitations. In *Proceedings of the Computer Security Foundations Workshop IV*, Franconia, NH, June 1991.

[GNY90] L. Gong, R. M. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, CA, 1990.

[Gro90] Adam Grove. Semantics for cryptographic protocols (early draft, sections 1-3 only). Manuscript, August 1990.

[GS90] K. Gaarder and E. Snekkenes. On the formal analysis of PKCS authentication protocols. In *Advances in Cryptology — AUSCRYPT '90 (LNCS 453)*, pages 106–121. Springer-Verlag, 1990.

[Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[HM84] Joseph Y. Halpern and Yoram Moses. Towards a theory of knowledge and ignorance. In *Proceedings of the 1984 AAAI Workshop on Non-monotonic logic*, pages 125–143, 1984. Reprinted in *Logics and Models of Concurrent Systems*, (ed. K. Apt), Springer-Verlag, 1985, pp. 459–476.

[HM90] Joseph Y. Halpern and Yoram Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, July 1990.

[KG91] Rajashekar Kailar and Virgil D. Gligor. On belief evolution in authentication protocols. University of Maryland, College Park, MD, February 1991.

[KNS90] J. Kohl, C. Neuman, and J. Steiner. The Kerberos network authentication service (version 5, draft 3). Available by anonymous ftp from athena-dist.mit.edu as /pub/doc/kerberos/V5DRAFT3-RFC.{PS,TXT}, October 1990.

[MNSS87] S. P. Miller, C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. Project Athena Technical Plan Section E.2.1, MIT, July 1987.

[MT88] Yoram Moses and Mark R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.

[Nes90] D. Nessett. A critique of the Burrows, Abadi and Needham logic. *Operating Systems Review*, 24(2), 1990.

[Rac89] Racal Research Ltd. Private communication from John Walker, 1989.

[SM89] Yoav Shoham and Yoram Moses. Belief as defeasible knowledge. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1168–1173, August 1989.

[Sne91] Einar Snekkenes. Exploring the BAN approach to protocol analysis. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1991. IEEE.

[Syv91] Paul Syverson. The use of logic in the analysis of cryptographic protocols. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 1991. IEEE.