A Unified Cryptographic Protocol Logic*

Paul F. Syverson
Code 5543
Naval Research Laboratory
Washington, DC 20375 USA
(syverson@itd.nrl.navy.mil)

Paul C. van Oorschot
Nortel Secure Networks
P.O. Box 3511, Station C
Ottawa, Canada K1Y 4H7
(paulv@nortel.ca)

November 1, 1996

Abstract

We present a logic for analyzing cryptographic protocols. This logic is based on a unification of four of its predecessors in the BAN family of logics, namely those given in [GNY90], [AT91], [vO93b], and BAN itself [BAN89]. The logic herein captures the desirable features of its predecessors and more; nonetheless, as a logic it is relatively simple and simple to use. We also present a model-theoretic semantics, and we prove soundness for the logic with respect to that semantics. We illustrate the logic by applying it to the Needham-Schroeder protocol, revealing that BAN analysis of it may lead to inappropriate conclusions in some settings. We also use the logic to analyze two key agreement protocols, examining an attack on one of them.

Introduction

In the late eighties Burrows, Abadi, and Needham developed BAN logic [BAN89], which quickly became the most widely used and widely discussed formal method for the analysis of identification/authentication protocols, particularly authenticated key distribution protocols. There have since been a number of papers noting BAN's inability or limited ability to reason about some features of both protocols and attacks on protocols. This has led several authors to propose alternatives to BAN. Many of these proposed alternatives are essentially extensions. These extensions yield an increase in reasoning power; however, collectively they accomplished this via a large number of linguistic and logical additions. As a result, one may be left unsure about the assumptions and meanings implicit in the application of these logics. Perhaps more significantly, one becomes increasingly unsure about the soundness of the reasoning that results. Relatedly, the simplicity that was part of BAN's basic appeal is lost.

This paper presents a logic that encompasses three of these logical expansions, those presented in [GNY90], [AT91], and [vO93b]. (Henceforth these logics will be referred to as 'GNY', 'AT', and 'VO', respectively.) And, since these are essentially expansions, this logic encompasses BAN itself as well. GNY and AT add to and reformulate BAN to better reason about the same class of protocols. VO adds rules to reason about key-agreement protocols. Our logic captures the desirable features of those logics. However, rather than simply tacking together the notation and rules from all of these we adopt an integrated approach, designed to yield a logic that is sound with respect to a single, relatively simple model of computation. Thus, this paper also presents a semantics underlying these logical expansions.[1] This will be of manifold advantage. First, some of these logics, including BAN itself, have been questioned before for lacking an independently motivated semantic foundation. (Cf., e.g., [Syv91].) Amongst other things, such a foundation can give us assurance that the reasoning in the logic is sound (i.e., false conclusions cannot be derived from true premises.) BAN was essentially given such a semantic foundation by Abadi and Tuttle in [AT91]. The model of computation and semantics herein is motivated by Abadi and Tuttle's but differs from it in fundamental ways. Second, having a fairly detailed model eliminates much of the confusion that can arise over the meaning of formal expressions and/or the applicability of logical rules. That is, since we can look at the semantic interpretation of an expression, we can make better decisions about whether that expression really says what we intend to say in a given circumstance. This helps in the protocol idealization step of a BAN or BAN-like analysis. (Analysis in this paper does not include idealization per se. More on this at the appropriate point.) Third, by serving as a common semantics, it allows us to view the extensions from a single perspective. Contrary to first appearances, this need not result in an overly complex logic. For, as a unifying model for comparison,

*Parts of this paper appeared in prelimary form in [vO93] and [SvO94].

[1] We refer here to a model theoretic semantics for a logic. This is not to be confused with a semantics for computer programs, which is generally any mathematical interpretation (formal or informal) of programming constructs.

it allows us to see what aspects of each logic can be captured by others and what not. There is thus a fair amount of syntactic reduction since primitives of one language are often definable in another. On the logical level there is a similar amount of axiom chopping. The result is a logic that is surprisingly simple.

In the next section of the paper we present a formal language and logic, and we describe the procedure whereby these are to be applied in protocol analysis. (Henceforth this logic will be called 'SVO'.) In §2, we give a basic description how to analyze protocols using the logic. We then analyze the well known Needham-Schroeder Protocol, henceforth 'NS', as an example [NS78]. This analysis demonstrates our analysis technique. It also allows us to compare our approach to that in [BAN89], in particular to examine a new observation, a misleading result that can be derived by using BAN analysis on the NS protocol. This highlights some of the advantages of SVO. In §3 we present a model of computation and a semantics for the language presented in §1, and we prove that the logic is sound with respect to the semantics. In §4 we apply SVO to two key agreement protocols, one from [MTI86] labelled 'A(0)', and the STS protocol from [DvOW92]. We derive that the protocols satisfy certain desirable goals and examine a potential attack on A(0). Finally, we present our conclusions and some directions for future work in §5.

The appendices give our arguments that SVO captures the expressive and deductive powers of GNY and VO. In appendix A we look at the language and logic of GNY in comparison to SVO. In appendix B we look at the language and logic of VO in comparison to SVO. In particular we consider in these sections how to capture in SVO the linguistic expressibility and logical derivability of GNY and VO. In so doing we also give definitions in SVO of useful expressions from the languages of those logics. We do not present a separate section for comparative discussions of AT. AT is the only previously given logic with a model-theoretic semantics. Comparisons between AT and SVO syntax require a semantic context as well, and, in the interest of brevity, we will not give a presentation of the full Abadi-Tuttle semantics. We therefore make comparative comments at appropriate points throughout §§1 and 3. (The rules and axioms of AT, GNY, and VO are summarized in appendices C–E for handy reference.)

1 Syntax

We will now present a logic capturing the desirable properties of BAN, AT, GNY, and VO that is both sound and relatively easy to use. Our presentation follows the structure of [AT91], with some important differences.

1.1 The Language

We begin with a definition of our language. Following Abadi and Tuttle, we reflect that we are looking at abstract protocols and are hence representing the sending of messages composed of expressions in a language rather than mere bitstrings. However, we expand the language slightly to cover, e.g., public keys, functions, and message comprehensibility. We also contract the language by doing away with separate syntax for forwarded messages and for binding messages to shared secrets. (The first is eliminated because we have no current use for it. The second is eliminated because its contributions are captured in our language by other means.)

We assume the existence of a set of primitive terms, $\mathcal{T}$, containing a number of sets of constant symbols representing principals, shared keys, public keys, private keys, numerical constants, etc. We also include a set of symbols, $\{*_1, *_2, \ldots\}$ to represent unrecognized received messages (or message fragments). We actually require two formal languages, one for messages and one for formulae. Only formulae can be true or false or have a principal's belief attributed to them. On the other hand, some messages are not formulae, e.g., a message consisting of a name and a nonce. In particular, no term is a formula, and vice versa. References to the language of SVO are meant to encompass both languages.

Messages and formulae of the language are built from $\mathcal{T}$ by mutual induction. The language of messages, $\mathcal{M}_{\mathcal{T}}$, is the smallest language over $\mathcal{T}$ satisfying:

- X is a message if $X \in \mathcal{T}$,
- $F(X_1, \ldots, X_n)$ is a message if $X_1, \ldots, X_n$ are messages and F is any function (including, e.g., ordered n-tuples, $(X_1, \ldots, X_n)$, encryptions, $\{X\}_K$, and signed messages $[X]_K$),
- φ is a message if φ is a formula.

The language of formulae, $\mathcal{F}_{\mathcal{T}}$, is the smallest language satisfying:

- $P \overset{K}{\leftrightarrow} Q$, $\mathrm{PK}_{\psi}(P, K)$, $\mathrm{PK}_{\sigma}(P, K)$, and $\mathrm{PK}_{\delta}(P, K)$ are formulae when P and Q are principals and K is a key.
- $\mathrm{SV}(X, K, Y)$ is a formula when X and Y are messages and K is a key.
- P *sees* X, P *received* X, P *says* X, P *said* X, and *fresh*(X) are formulae when X is a message and P is a principal,
- $\neg\varphi$ (not-φ) and $\varphi \wedge \psi$ (φ and ψ) are formulae if φ and ψ are formulae (other connectives are definable in the usual manner)[2],

[2] We will use '$\supset$' (pronounced "horseshoe") rather than '$\rightarrow$' to represent the conditional to avoid confusion with the standard notation for sending a message in protocol description, e.g., '$A \longrightarrow B$'. (In our primitive notation, $\varphi \supset \psi$ is of course $\neg\varphi \vee \psi$ [Men87].)

- *P believes* φ and *P controls* φ are formulae when φ is a formula and P is a principal.

Most of the expressions just given either are standard usage in BAN and its derivatives or should be intuitively clear. We give a brief intuitive description here for those that may not be. '*P controls* φ' indicates that P is a trusted authority on φ. If P says φ, then φ is so. '$P \overset{K}{\leftrightarrow} Q$' indicates that K is a symmetric key shared exclusively by P and Q. No one other than P or Q will ever encrypt messages using K, and only P, Q, and those they trust (e.g., a server who might generate it) know K. '$\mathrm{PK}(P, K)$' is used similarly for public keys. K is P's public key, and 'K^{-1}' is used exclusively to refer to the corresponding private key. '$\mathrm{PK}_\psi(P, K)$', '$\mathrm{PK}_\sigma(P, K)$', and '$\mathrm{PK}_\delta(P, K)$' are for encryption, signature, and key agreement keys, respectively. Keys themselves may or may not have subscripts.[3] Typically, keys and nonces have mnemonic subscripts, e.g., $A \overset{K_{ab}}{\longleftrightarrow} B$. '$\mathrm{SV}(X, K, Y)$' refers to signature verification. It says that, given signed message X, applying K to it as a signature verification key verifies Y as the message signed with the corresponding private key. The meaning in our semantics of all expressions will be discussed below in §3.2.

A few more notes on notation: Typically '$\{X\}_K$' is meant to refer to transformations of X using K. We mean specifically to include shared and public key encryption under this notation. We find the following notation useful for giving a uniform presentation of the axioms. $\widetilde{K}$ is the *complement* of key K. In public key ciphering schemes, K^{-1} is the complement of K, and K is the complement of K^{-1}. In shared key schemes $K = \widetilde{K}$. Unless restricted, either explicitly or implicitly by context, 'K' will refer below to any symmetric, private, or public key. We can always treat encryption and decryption as functions parameterized by the relevant key. Thus, we can generalize this notation to '$\widetilde{F}$', expressing the complement of a function F. This notation assumes that we are referring to an effectively one-one (injective) function, that is, a function such that it is computationally difficult to find pairs of arguments mapping to the same value, whether or not that value is given. It does not assume that either the function or its complement (inverse) is feasibly computable in practice.

Some previous BAN logics have used expressions such as '$\{X\}_K$' to represent digital signatures as well as encryptions. If one uses simple RSA exponentiation with a private key for signatures, then it is possible to treat a digital signature as simply the inverse of public key enciphering. Thus, given the public key, $\widetilde{K}$, one can recover X from $\{X\}_K$, and the notational choice is somewhat natural. We instead use '$[X]_K$' to represent message X digitally signed using key K. In most modern signature schemes it is not possible to recover X from the signature itself, even if one possesses $\widetilde{K}$. Thus, signing is not in any reasonable sense the inverse of encryption. To make clear that we are assuming a standard signature scheme (without message recovery) we have adopted this notation. '$[X]_K$' refers to the signed message, not just the signature. Therefore, anyone in possession of $[X]_K$ is automatically in possession of X.

Throughout the paper φ and ψ are metalinguistic symbols used to refer to arbitrary formulae. Γ is a metalinguistic symbol referring to sets of formulae.

1.2 The SVO Logic

Our logic is a modal logic [Che80]. It has two inference rules:

Modus Ponens: From φ and $\varphi \supset \psi$ infer ψ.

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

'$\vdash$' is a metalingusitic symbol.[4] '$\Gamma \vdash \varphi$' means that φ is derivable from the set of formulae Γ (and the axioms as stated below). '$\vdash \varphi$' means that φ is a theorem, i.e., derivable from axioms alone. We describe derivability (i.e. proofs) below in §2. Axioms are all instances of tautologies of classical propositional calculus [Men87], and all instances of the following axiom schemata[5]:

Believing For any principal P and formulae φ and ψ,

Ax1. P *believes* $\varphi \wedge P$ *believes* $(\varphi \supset \psi) \supset P$ *believes* ψ

Ax2. P *believes* $\varphi \supset P$ *believes* $(P$ *believes* $\varphi)$

Axiom Ax1 says that a principal believes all that logically follows from his beliefs. Axiom Ax2 says in effect that a principal can tell what he believes.

Source Association Keys are used to deduce the identity of the sender of a message.

Ax3. $(P \overset{K}{\leftrightarrow} Q \wedge R$ *received* $\{X^Q\}_K) \supset$
$(Q$ *said* $X \wedge Q$ *sees* $K)$

Ax4. $(\mathrm{PK}_\sigma(Q, K) \wedge R$ *received* $X \wedge \mathrm{SV}(X, K, Y)) \supset$
Q *said* Y

---

[3] In [BAN89], the public key and shared key notations for indicating key appropriateness were more similar. We have followed the notational conventions of [GS91] and [vO93b]. In the presence of three types of public keys, we find this to be the best compromise between familiarity and readibility. Further issues that lead to this choice of public key notation are discussed in appendix B.

[4] The symbol '$\vdash$' is usually pronounced "turnstile". The symbol '$\models$', to be introduced, is pronounced "double turnstile".

[5] Some of the following are proper axioms, logically. Those containing metavariables for formulae are actually axiom schemata. We will generally ignore this distinction, referring to all as 'axioms'.

Recall that '$\mathrm{PK}_\sigma(Q,K)$' says that K is the public signature verification key for Q, and '$\mathrm{SV}(X,K,Y)$' says that given signed message X, applying K to it as a signature verification key verifies Y as the message signed with $\widetilde{K}$. In saying '$\mathrm{PK}_\sigma(Q,K)$' we assume enough redundancy in the signature scheme to preclude attackers possessing only K from producing a valid signature for Q on any message, meaningful or otherwise. This feature is designed into most modern signature schemes. Precise meaning is set out in §3.2.

By definition, all symbols in the axioms are symbols of the languages specified above, $\mathcal{F}_\mathcal{T}$ and $\mathcal{M}_\mathcal{T}$. Thus, in particular, the X in these axioms is a message not a bitstring. But, a key can be applied (by anyone who has it) to any bitstring to yield another bitstring. This apparent incongruity is handled in our language by the unrecognized message symbols $\{*_1, *_2, \ldots\}$, which will be discussed more below. The superscripted Q in axiom Ax3 indicates that the message is from Q (rather than P). Honest principals possessing the key K are assumed to be able to correctly indicate message origin, and others possessing K are assumed to be able to evaluate correctly indicated message origin. Principals not possessing K are assumed to be unable to so indicate or evaluate origin. (This notation is admittedly an inelegance. There is no standard mechanism to indicate who a message is from. Even if a message contains an encrypted *who_from* field there is no standard place it need occur in an authentication protocol message. Further, some contextual mechanisms do not explicitly indicate the sender at all. For example, consider the handshake at the end of the Needham-Schroeder protocol, discussed in §2.1. Leaving redundancy issues raised in §2.1 aside, message 4 indicates that it is from B simply by being the first use of the distributed key K_{ab}. Message 5 is indicated to be from A by varying the (unpredictable) plaintext contents of message 4 in a predictable way and then reencrypting with K_{ab}. Whether such mechanisms are appropriate in context to justify use of the superscript notation is something that should be evaluated extralogically.)

Key Agreement Session keys that are the result of good key-agreement keys are good.

Ax5. $((\mathrm{PK}_\delta(P,K_p)) \wedge (\mathrm{PK}_\delta(Q,K_q))) \supset P \stackrel{F_0(K_p,K_q)}{\longleftrightarrow} Q$

Ax6. $\varphi \equiv \varphi[F_0(K,K')/F_0(K',K)]$

Recall that '$\mathrm{PK}_\delta(R,K)$' says that K is the public key-agreement key for R and implies that K^{-1} remains secret. Precise meaning is set out in §3.2. Here '$F_0(K_p,K_q)$' implicitly indicates a key agreement function as in Diffie-Hellman key exchange [DH76]. The indication is implicit because the explicit arguments of F_0 are both public keys. Key agreement is a function of one public and one private key. The function implied by F_0 is a key agreement function using the public key in the first argument of F_0 with the private key corresponding to the second argument. In Ax6 '$\varphi[F_0(K,K')/F_0(K',K)]$' indicates the same formula as φ except that $F_0(K,K')$ is substituted everywhere that $F_0(K',K)$ occurs in φ. The axiom says that the two formulae are logically equivalent. In other words, the logic respects the symmetry of key agreement.

Receiving A principal receives the concatenates of received messages and decryptions with available keys, as well as the message contained in any received signed message.

Ax7. P *received* $(X_1,\ldots,X_n) \supset P$ *received* X_i

Ax8. (P *received* $\{X\}_K \wedge P$ *sees* $\widetilde{K}) \supset P$ *received* X

Ax9. P *received* $[X]_K \supset P$ *received* X

Seeing A principal sees anything he receives. A principal also sees all components of every message he sees and any message he can compute from what he sees. The difference in meaning between seeing and receiving is made precise in §3.2.

Ax10. P *received* $X \supset P$ *sees* X

Ax11. P *sees* $(X_1,\ldots,X_n) \supset P$ *sees* X_i

Ax12. (P *sees* $X_1 \wedge \ldots \wedge P$ *sees* $X_n) \supset$
(P *sees* $F(X_1,\ldots,X_n)$)

Here F is meta-notation for any function feasibly computable in practice by P, for example, $X_1 + \ldots + X_n$. There is no axiom for seeing corresponding to axiom Ax8 for receiving, i.e., (P *sees* $\{X\}_K \wedge P$ *sees* $\widetilde{K}) \supset$ P *sees* X. Such an axiom is a special case of axiom Ax12, where F is the application of $\widetilde{K}$ to $\{X\}_K$.

Comprehending If a principal comprehends a message and sees a function of it (of the appropriate type), then he understands that this is what he is seeing.

Ax13. P *believes* (P *sees* $F(X)) \supset P$ *believes* (P *sees* X)

Here F is meta-notation for any effectively one-one function such that either F or $\widetilde{F}$ is computable in practice by P. F may represent encryption or decryption where the relevant key is treated as a parameter.

This axiom is fairly subtle. It might appear to imply that P can invert F, i.e., can readily find X given the value of $F(X)$. Actually, if P can calculate F but not invert it, then axiom Ax13 says that he only knows he has $F(X)$ if he already knows that he has X. This axiom captures what we want of GNY's recognizability. Note that the converse of axiom Ax13 is a theorem, following from axiom Ax1 and axiom Ax12 by necessitation and modus ponens.

Saying A principal who has said a concatenated message has also said and sees the concatenates of that message. A principal who has recently said X has said X. A principal sees what he says.

Ax14. P *said* $(X_1, \ldots, X_n) \supset (P$ *said* $X_i \wedge P$ *sees* $X_i)$

Ax15. P *says* $(X_1, \ldots, X_n) \supset$
$(P$ *said* $(X_1, \ldots, X_n) \wedge P$ *says* $X_i)$

Jurisdiction This axiom in effect says that P's word is law for the φ in question.

Ax16. $(P$ *controls* $\varphi \wedge P$ *says* $\varphi) \supset \varphi$

Freshness A concatenated message is fresh if one of its concatenates is fresh, and any effectively one-one function F (including encryption and decryption) of a fresh message is fresh.

Ax17. *fresh*$(X_i) \supset$ *fresh*$(X_1, \ldots, X_n)$

Ax18. *fresh*$(X_1, \ldots, X_n) \supset$ *fresh*$(F(X_1, \ldots, X_n))$

The function F in axiom Ax18 must be genuinely dependent on the fresh component. For example, if X_2 is fresh, then (X_1, X_2, X_3) is fresh; however, the value of $X_1 + (0 \cdot X_2) + X_3$ is not. Specifically, a function is *genuinely dependent* on an argument if computing the value of the function is infeasible without the value of that argument.

Nonce-Verification Freshness promotes a message from having been said (sometime) to having been said during the current epoch.

Ax19. $($*fresh*$(X) \wedge P$ *said* $X) \supset P$ *says* X

Symmetric goodness of shared keys A shared key is good for P and Q iff it is is good for Q and P.

Ax20. $P \overset{K}{\leftrightarrow} Q \equiv Q \overset{K}{\leftrightarrow} P$

2 Protocol Analysis

In this section we give a brief description of our syntactic protocol analysis technique, which is somewhat similar to the techniques associated with previous BAN logics. A major difference is that we do not idealize the protocol. (What 'idealize' means will be explained in the next subsection.)

Syntactic analysis comes in two main steps. First, we set out premises that reflect assumptions based on the protocol description. Second, we prove desired goals using those premises together with the axioms and rules of the logic. These steps are typically carried out against a background of goals the protocol is intended to meet. Should we fail to prove one or more of these goals, we may want to add the step of considering why the protocol fails to meet its goals. This may include looking for possible attacks. Relatedly, we may semantically analyze our premises to see if any of them can be false in a run of the protocol. (Semantics is discussed below in §3.)

Premises can typically be grouped into four types. First are initial assumptions, those things assumed to be true at the start of the protocol. Examples include each principal's belief in the freshness of nonces it generates, the goodness of long term keys principals share with servers, the jurisdiction of a server over the quality and freshness of keys it sends, etc. Also included are premises reflecting a principal's comprehension of terms it simply has without receiving them during the current protocol run and premises reflecting a principal's comprehension of relevant signature verifications. For a specific example, consider a protocol step in which a key server S distributes a key to principal A for the purpose of talking with B:

$S \longrightarrow A$: $\{T_s, B, K_{ab}\}_{K_{as}}$

This means that S has sent the following to A (all encrypted with K_{as}, a symmetric key shared by A and S): a timestamp, T_s, B's identifier, and the session key K_{ab}. Premises of the first type that would play a role in analyzing this message would include *fresh*(T_s), A *believes fresh*(T_s), $A \overset{K_{ab}}{\longleftrightarrow} B$, and A *believes* $A \overset{K_{as}}{\longleftrightarrow} S$.

Second are premises reflecting the receipt of messages sent in a protocol run. These can be taken directly from the protocol specification. The corresponding reception premise for the protocol step just presented would be

$$A \textit{ received } \{T_s, B, K_{ab}\}_{K_{as}}.$$

These are often unused in proofs, but they help in the formation of later premises.

Third are premises reflecting what is comprehended by each principal of the messages he receives. Even if A receives $\{T_s, B, K_{ab}\}_{K_{as}}$, she might not understand all of the message. For example, the random nature of distributed keys makes them inherently unrecognizable (in themselves by those who did not generate them). Assuming timestamps and names are recognizable, a premise of this type corresponding to the above protocol step would be

$$A \textit{ believes } A \textit{ received } \{T_s, B, *_s\}_{K_{as}}$$

. In practice, it is generally clear how to produce such premises from the premises of the second type in consideration of the submessages that are comprehended by the receiving principal.

Fourth are premises reflecting the interpretation that a receiver attaches to a received message. (This is the primary replacement for idealization.) These indicate what the receiver assumes the sender meant by a given message. For the above protocol step, a reasonable candidate would be

$$\begin{aligned} &A \textit{ believes } (A \textit{ received } \{T_s, B, *_s\}_{K_{as}} \supset \\ &\quad A \textit{ received } \{T_s, B, A \xleftrightarrow{K_{ab}} B, \textit{fresh}(K_{ab})\}_{K_{as}}) \end{aligned}$$

This is only a candidate since the actual premise appropriate for this protocol might depend on features of the protocol that our simple example does not capture, such as other messages A has sent or received. We will be analyzing an actual protocol presently and will then further illustrate and discuss premises of various types.

With the premise set established we attempt to derive various goals concerning the protocol. A *proof* is a sequence of formulae in the logic. Each line of a proof is either a premise, an axiom, or derivable from preceding lines via modus ponens or necessitation. Our notion of proof differs from Abadi and Tuttle's since they only allow modus ponens to apply to theorems of the logic. This would preclude premises as legitimate lines in a proof.[6]

In AT and SVO necessitation must always be restricted to theorems. This is a crucial point about proofs, which may be missed by those unfamilar with logic per se. Theorems are formulae provable from axioms alone. The rule of necessitation cannot be applied to any of the above premise examples nor to any line in a proof that depends on a premise. Otherwise we could use necesitation to show that any principal believes anything that we have assumed. This is a mistake, even if what we have assumed is true. For example, suppose that $A \xleftrightarrow{K_{ab}} B$ is true. We do not want to therefore conclude that C *believes* $A \xleftrightarrow{K_{ab}} B$.

Syntactic analysis of the type just described is all that is available using BAN, GNY, and other logics without an independent semantics. AT and SVO add another level to this by providing an independently motivated model-theoretic semantics. In addition to other values, this allows one to do semantic analysis of the protocol. One advantage of this is a rigorous means of assessing the truth of initial assumptions and other premises. Problems arising from initial assumptions, as in the Nessett protocol [Nes90], are thus addressible using these logics. (Cf. [Syv92] for a detailed discussion.) We now look at a specific example to illustrate our analysis technique.

[6] Our choice to characterize proofs in this way has important repercussions for other features of the logic. In AT, since every line of a derivation must be a theorem of the logic, it is necessary for analysis to restrict consideration to "good" runs where, e.g., negations do not occur within belief operators in initially held beliefs. We need place no such restrictions. (These restrictions are not present in [AT91] simply to make derivations sound; they have other motivations as well, which we will not discuss.)

2.1 The Needham-Schroeder Protocol

NS is a typical protocol for key distribution to two principals via an on-line authentication and key distribution server [NS78]. It is also a standard example for analysis because it is subject to an attack that has long been well known [DS81]. The protocol is as follows:

$$\begin{array}{ll} 1.\ A \longrightarrow S: & A, B, N_a \\ 2.\ S \longrightarrow A: & \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \\ 3.\ A \longrightarrow B: & \{K_{ab}, A\}_{K_{bs}} \\ 4.\ B \longrightarrow A: & \{N_b\}_{K_{ab}} \\ 5.\ A \longrightarrow B: & \{N_b - 1\}_{K_{ab}} \end{array}$$

In the first message A tells the server that she would like to obtain a session key for talking with B, and she includes a nonce, N_a, for S to include in his response thus identifying it as a response. In the second message, S sends A the session key, K_{ab}, B's name indicating that it is for a session with B, and A's nonce. He also includes a message encrypted with a key S shares with B consisting of the session key and A's name to show that the key is for talking with A. The whole second message is encrypted with a key that S shares with A. A decrypts the message, and, if the nonce and B's name agree with the message she sent, she forwards the portion encrypted for B to B in message 3. B decrypts this to obtain the session key. He then generates a nonce and encrypts it with K_{ab} and sends this to A. A decrypts the nonce and subtracts one from it (to distinguish the final message from a simple reflection of message 4, which could be from anyone). She then encrypts this with K_{ab} and sends it to B.

2.2 Analysis of the NS protocol

The first step in analyzing the protocol is to set out the assumptions that we make based on the protocol specification. These will serve as premises, which we will use together with the axioms and rules of the logic to derive conclusions. Generic assumptions include each principal's belief that the nonces it generates are fresh, belief that the server has jurisdiction over the freshness and goodness of session keys it sends, and belief that the long term key it shares with the server is good. Formally, these are

P1 A *believes fresh*(N_a)
B *believes fresh*(N_b)

P2 A *believes* S *controls* $(A \xleftrightarrow{K_{ab}} B)$
B *believes* S *controls* $(A \xleftrightarrow{K_{ab}} B)$

P3 A *believes* S *controls* (*fresh*(K_{ab}))
B *believes* S *controls* (*fresh*(K_{ab}))

P4 A *believes* $(A \xleftrightarrow{K_{as}} S)$
B *believes* $(B \xleftrightarrow{K_{bs}} S)$

We also assume that each of the principals received the messages they were sent. (Since we do not use the first message in our analysis, we do not bother with a corresponding premise.)

P5 A *received* $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$

P6 B *received* $\{K_{ab}, A\}_{K_{bs}}$

P7 A *received* $\{N_b\}_{K_{ab}}$

P8 B *received* $\{N_b - 1\}_{K_{ab}}$

Received messages are not necessarily understood. We must explictly include in the premise set what messages are understood by the principals and what those messages mean. Thus, we include A *believes* A *received* X for each message X that A is assumed to comprehend based on redundancy or an expectation, e.g., a nonce A sent out—and similarly for B. (We have not included any premise corresponding to **P7** (message 4) since there is nothing in that message that is comprehensible to A.)

P9 A *believes* A *received* $\{N_a, B, *_1, *_2\}_{K_{as}}$

P10 B *believes* B *received* $\{*_3, A\}_{K_{bs}}$

P11 B *believes* B *received* $\{N_b - 1\}_{*_3}$

Finally, we include premises corresponding to the assumed meaning that principals attach to received messages. (These correspond to the assumptions implicit in idealizing a protocol as in [BAN89].)

P12 A *believes* (A *received* $\{N_a, B, *_1, *_2\}_{K_{as}} \supset$
A *received* $\{N_a, B, A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), *_2\}_{K_{as}}$)

P13 B *believes* (B *received* $\{*_3, A\}_{K_{bs}} \supset$
B *received* $\{A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab})\}_{K_{bs}}$)

P14 B *believes* ((B *received* $\{*_3, A\}_{K_{bs}} \wedge$
B *received* $\{N_b - 1\}_{*_3}$) $\supset$
B *received* $\{N_b - 1\}_{K_{ab}}$)

These premises preclude automated analysis because they typically vary from protocol to protocol even for a message with the same specification. Mao and Boyd have a BAN-like formal method that does allow for full automation [MB93]. They accomplish this by requiring that the protocol be specified in their own language, at a much greater level of detail than usual. In a sense, they thus incorporate the idealization into the specification. GNY does something similar in its message interpretation rules.

Note that, in **P14**, for B to believe he has received $\{N_b - 1\}_{K_{ab}}$ it is not enough that he receive the message that he interprets to say this; he must also believe he has received the previous message in which S told him K_{ab}. Without the previous message, he would not have the key and could not recognize it as a (candidate) key for speaking with A.

We can now proceed with our formal derivation of goals using SVO. In the interests of brevity, we will compress many of the proof steps together, and we will not cite the use of propositional reasoning used in giving the justification for derivation lines. The first derivation is of goals for A. We will discuss some typical goals for protocols in §4.1. The goals we derive here are that A believes the distributed key is good for talking with B (line 5) and that A believes the distributed key is fresh (line 6). In the justification of each line in any derivation, 'Axn' refers to axiom Axn of our logic, 'Nec' to the Necessitation rule, and 'MP' to the Modus Ponens rule.

1. A *believes*
   A *received* $\{N_a, B, A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), *_2\}_{K_{as}}$
   by P9, P12, Ax1, MP

2. A *believes* S *said* $(N_a, B, A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), *_2)$
   by 1, P4, Ax3, Ax1, Nec, MP

3. A *believes* $\mathit{fresh}(N_a, B, A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), *_2)$
   by P1, Ax17, Ax1, Nec, MP

4. A *believes* S *says* $(N_a, B, A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), *_2)$
   by 2, 3, Ax19, Ax1, Nec, MP

5. A *believes* $A \xleftrightarrow{K_{ab}} B$
   by 4, P2, Ax15, Ax16, Ax1, MP

6. A *believes* $\mathit{fresh}(K_{ab})$
   by 4, P3, Ax15, Ax16, Ax1, MP

We now derive goals for B. These are rather different than for A. B can only conclude that S once said that the key is is good for talking to A and that it is fresh (lines 3 and 4 below). He cannot conclude that the key is good or fresh. He can also confirm the existence of some currently active, far end party who knows the key (line 5 below).

1. B *believes* B *received* $\{A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), A\}_{K_{bs}}$
   by P10, P13, Ax1, MP

2. B *believes* S *said* $(A \xleftrightarrow{K_{ab}} B, \mathit{fresh}(K_{ab}), A)$
   by 1, P4, Ax3, Ax1, Nec, MP

3. B *believes* S *said* $A \xleftrightarrow{K_{ab}} B$
   by 2, Ax14, Ax1, Nec, MP

4. B *believes* S *said* $\mathit{fresh}(K_{ab})$
   by 2, Ax14, Ax1, Nec, MP

5. B *believes* B *received* $\{N_b - 1\}_{K_{ab}}$
   by P10, P11, P14, Ax1, MP

2.2.1 SVO vs. BAN analysis of the NS protocol

We now discuss the results of our analysis and contrast them with those of the analysis of NS in [BAN89]. We feel that the above analysis is about as simple as the one in [BAN89]. While there can be no objective measure of this, we emphasize that the proofs in [BAN89] are sketchier than the above. This may lead to an appearance of greater simplicity. We now turn to the premises of each analysis.

P1–P4 constitute a subset of the assumptions given in [BAN89]. The BAN assumptions also include assumptions about the server's belief in the quality of the long term keys and the quality and freshness of distributed session keys. While reasonable, they are unnecessary for the analysis given in [BAN89] or the one herein, so we have left them out. It is interesting to note that, even if unnecessary, these assumptions are more natural in the context of BAN analysis since it is there necessary to derive that A believes S believes $A \stackrel{K_{ab}}{\longleftrightarrow} B$ in order to derive A *believes* $A \stackrel{K_{ab}}{\longleftrightarrow} B$. These assumptions thus attest to the consistency of such a second order belief with the first order belief that is its object. In other words, if these assumptions of first order belief are true, then the corresponding second order beliefs cannot be incorrect. In our analysis, this second order belief is replaced with the more conservative formula A *believes* S *says* $A \stackrel{K_{ab}}{\longleftrightarrow} B$. Nonetheless, note that assumptions such as **P2**, which are common in such analysis, can be somewhat dangerous [vO93a].

The assumptions in [BAN89] also include the assumption that B believes the session key to be fresh. As noted by Burrows et al., this is a dubious assumption that overlooks the possibility of attacks in which an old, compromised session key is used, such as in the Denning-Sacco attack. It is included in [BAN89] not because the authors think it is justifiable, but to illustrate that a desirable protocol goal cannot be reached without it.

P5–P8 reflect the messages that each principal receives. These directly reflect the specified protocol without any interpretation of message contents, as would occur in idealization. They correspond to premises based on the protocol annotation step of analysis in [BAN89]; although in a BAN analysis, protocols are annotated only after idealization. As would typically be the case, these premises play no role in our proofs; however, they do play a role in our analysis by helping us to see what the following premises should look like.

P9–P11 do not directly correspond to anything in BAN analysis, except perhaps to global assumptions about the recognizability of messages. They reflect which parts of received messages receivers can tie back to originally understood message components or to each other.

P12–P14 reflect how receivers interpret received messages in the context of the given protocol. They are typically the hardest premises to produce, sometimes requiring awareness of intended application and context in addition to the protocol speicification. These correspond roughly to idealization and annotation in [BAN89]. There, idealization interprets the meaning of messages, and annotation allows the assumption of a premise expressing that the receiver received the idealized message he was sent. The BAN approach is typically a little less explicit. This lack of explicitness naturally engenders less detail of expression (hence greater simplicity in appearance).

The idealization of NS from [BAN89], expressed in our notation, is as follows:

$$
\begin{array}{lll}
2. & S \longrightarrow A: & \{N_a, (A \stackrel{K_{ab}}{\longleftrightarrow} B), \mathit{fresh}(K_{ab}), \\
 & & \{A \stackrel{K_{ab}}{\longleftrightarrow} B\}_{K_{bs}}\}_{K_{as}} \\
3. & A \longrightarrow B: & \{A \stackrel{K_{ab}}{\longleftrightarrow} B\}_{K_{bs}} \\
4. & B \longrightarrow A: & \{N_b, (A \stackrel{K_{ab}}{\longleftrightarrow} B)\}_{K_{ab}} \textit{ from } B \\
5. & A \longrightarrow B: & \{N_b, (A \stackrel{K_{ab}}{\longleftrightarrow} B)\}_{K_{ab}} \textit{ from } A
\end{array}
$$

We will now illustrate the significance of one important difference between our receiver's interpretation premises and BAN's idealization and annotation in the context of what is provable from them.

Derived Goals of the analysis in [BAN89] include that A believes that B believes $A \stackrel{K_{ab}}{\longleftrightarrow} B$. Nothing comparable is provable in the above analysis. Since this is a desirable result, the above analysis might be too weak. Whence the difference?

Idealization attaches one meaning for both the sender and the receiver; whereas receiver's interpretation premises attach only the meaning for the receiver. Thus, in [BAN89], the inclusion of $A \stackrel{K_{ab}}{\longleftrightarrow} B$ in the idealization of messages 4 and 5 is to assure "each principal that the other believes the key is good. These statements can be included because neither message would have been sent if the statements were not believed." [BAN89], p. 19. The inclusion is thus based on the intended meaning of a message on the part of the sender. However, BAN annotation based on this idealization allows us to derive that A believes that B once said $A \stackrel{K_{ab}}{\longleftrightarrow} B$. And, based on this we can prove that A believes that B believes $A \stackrel{K_{ab}}{\longleftrightarrow} B$. In other words, BAN analysis allows us to prove things about the receiver's interpretation of a message based on the interpretation intended by the sender. Unfortunately, it is possible to slip an attack between these two interpretations.

We hasten to emphasize that what we are about to look at is not an attack on NS per se. Rather it is an analysis of NS that shows it is incorrect to conclude based on the specification that A believes that B believes $A \stackrel{K_{ab}}{\longleftrightarrow} B$. There is nothing in [NS78] to indicate that NS was meant to achieve mutual belief in shared keys or even entity authentication of B to A. (It was intended to achieve key freshness for B via entity authentication of A to B, and, if we assume session keys can be obtained by adversaries within the lifetime of long term keys, then it did not succeed in this [DS81].) Thus the following reveals a limitation of the BAN analysis technique, rather than a flaw in the NS protocol.

Suppose that the NS protocol runs properly through the sending of the third message, but an attacker intercepts this message so it is never received by B. In place of message 4, the attacker simply sends a random string of the appropriate length. A then 'decrypts' this string using K_{ab}. Since there is nothing in the message that is recognizable to A, she cannot tell whether the result is a nonce sent by B. So, she subtracts one from the result, reencrypts it with K_{ab}, and sends it to B as message 5. This is also intercepted by the attacker.

According to the analysis in [BAN89], after a run of NS A believes that B has expressed faith in the quality of K_{ab}. But, in this attack B is not even present. Thus, the derivation is misleading with respect to both entity authentication and mutual belief in the quality of a shared key. This attack is much easier to implement than the Denning-Sacco attack since it does not require any key compromise in order to succeed. As already noted, however, it is not an attack on intended protocol goals. We also hasten to note that it falls explicitly outside the scope of the analyses in [BAN89]. In [BAN89], there is a blanket assumption that "[e]ach encrypted message contains sufficient redundancy to allow a principal who decrypts it to verify that he has used the right key." (pp. 5–6) There is thus no flaw in the analysis of NS therein.

Further, the blanket assumption is frequently, if not universally, a reasonable assumption to make. In particular, the attack would not be possible in many modern implementations of the protocol. In practice encryption often contains a mechanism to check that when decrypted the correct decryption key was used, for example, including a hash of the message content along with that content inside the encryption. This is not represented in the NS protocol specification; though it would be trivial to do so. Even though the blanket assumption pushes protection against such attacks outside the scope of [BAN89], it is certainly possible to represent the protection mechanisms in question at the specification level of [BAN89]. And, there are protocols for which it is inadvisable to include the redundancy generally assumed in [BAN89]. (For example, cf. [BM92, BM93].) Thus, it is better to represent such mechanisms in the specification whenever they are actually intended.

We have focussed on equating sender's and receiver's meaning in a BAN analysis as opposed to an SVO analysis. There is another feature of SVO analysis that is equally important to uncovering the limited applicability of NS for entity authentication of B to A: our requirement that premises explicitly set out what principals comprehend. This immediately brought out that A does not comprehend N_b in this protocol. Thus, a result showing that A understood anything by receiving message 4 would have to be incorrect.

Note also that our logical derivations do not themselves lead to our discovery. Rather we are only able to prove limited results because the relevant premises make assumptions only about a receiver's interpretation of a message. The inability to prove desired goals in this case is one factor in uncovering the inapplicability of NS for entity authentication of B to A. As noted by the philosopher of mathematics Imre Lakatos, sometimes the virtue of logical proof is not that it compels belief but that it suggests doubt. (We discuss some typical goals that protocols might be intended to meet in §4.1.)

Finally, though we have entirely replaced idealization, we do not claim to have removed the possiblility of error in interpreting the meaning of messages. What we have replaced idealization with is further assumptions (premises) for each protocol. And, though the latter may seem more complicated, we are simply being explicit where analogous reasoning was done implicitly in BAN analysis. It is still possible to incorrectly assume that receipt of a given message in a given context implies that a certain content has been received. Relatedly, our model-theoretic semantics can be used to rigorously, albeit informally, evaluate the truth of all premises.

3 Semantics for SVO

3.1 Model of Computation

Computation is performed by a finite set of principals, $P_1, \ldots, P_n$, who send messages to one another. In addition there is a principal P_e representing the environment. This allows modelling of any penetrator actions as well as reflecting messages in transit.

Each principal P_i has a local state s_i. A global state is thus an $(n+1)$-tuple of local states. Principals can perform three actions: sending a message, receiving a message, and generating new data, such as keys. These are denoted by $send(X, G)$, $receive()$, and $generate(X)$ respectively. One can send and receive any message, but one can only generate primitive terms, i.e., members of $\mathcal{T}$. Other than generating new data, internal computations are not represented as actions. They are represented implicitly. Each action produces a transition from one state to the next. Note that receiving is an action, performed by the principal P_i who receives a message. The action itself is viewed as the nondeterministic choice of some message from P_i's buffer. This is why it is listed as having no argument. Once performed, however, the resulting local state reflects which message was received, e.g., $receive(X)$. Sending is always directed to a set of principals, G. If only one principal is the intended recipient, G is a singleton. If a message is indiscriminantly broadcast, G is the set of all principals.

A run is an infinite sequence of global states indexed by integral times. The first state of a given run r is assigned a time $t_r \leq 0$. The initial state of the current authentication is at $t = 0$. The global state at time t in run r is $r(t)$, and the corresponding projection to P_i's local state is $r_i(t)$. We may also write $r(t)$ as '(r,t)'. We will also occasionally refer to global states thus represented as points or (possible) worlds. (Cf. §3.2 under **Believing**.)

The local state of each principal includes a local history of all the actions the principal has performed up to that point and a set of available transformations. These are the computations that are feasibly computable by that principal. Typically, for a given principal, P_i, the available transformations, A_i, consists of arbitrary numbers of applications of the message formation rules (including term formation rules and formula formation rules) in the definition of the language of messages, $\mathcal{M}_{\mathcal{T}}$, up to the computational complexity limitations of that principal. These include encryptions and decryptions with available keys as well as other functions the principal may perform, e.g., hashes, signatures, arithmetical functions, etc. While the number of messages known to a principal may increase over time, his computational ability to form new messages, i.e. A_i, is assumed to stay fixed. All principals are assumed to be able to decide the equality of any messages they can produce from what they have. For example suppose a public key ciphering scheme is being used in which encryption and decryption are commutative, such as RSA. If P_i has message X and K_j, the public encryption key of P_j, then P_i can verify, given $\{X\}_{K_j^{-1}}$, that $X = \{\{X\}_{K_j^{-1}}\}_{K_j}$ even if he cannot form $\{X\}_{K_j^{-1}}$.

The environment's state consists of a global history, a set of transformations available to the environment, and a message buffer m_i for messages sent to P_i and not yet received. We limit the set of runs to those where a given message can only be received after it is sent. Thus, if $\mathit{receive}(X)$ is in the local history at $r_i(t)$, then $\mathit{send}(X,G)$ is in the local history at some $r_j(t')$, where $t' < t$.

As mentioned, transformations on a message are implicitly made when that message is sent or received. For example, if a principal receives an encrypted message $\{X\}_K$ and he has $\widetilde{K}$, then he has also received X. Specifically, the set of *explicitly received messages* for a principal P_i at a point (r,t) contains the following: (1) all messages X such that $\mathit{receive}(X)$ appears in the local message history at or prior to t, (2) the concatenates of any concatenated received message, (3) any message X for which $\{X\}_K$ is a received message and appropriate application of $\widetilde{K}$ is an available transformation for P_i, and (4) any message X for which $[X]_K$ is a received message for some K. Note that under this definition, if P_i receives an encrypted message and later acquires the decryption key, the decryption is a received message at that later point in the run.

For a given principal P_i, the collection of all messages that are explicitly received, newly generated, or initially available to P_i implicitly defines a set of *seen messages* for him at that point.[7] This consists of the messages just mentioned plus all the messages he can recursively produce from those messages via his available transformations (up to the limits of his computational capabilities).

Rather than being explicit, some received messages are highly contextual. For example, we saw in the Needham-Schroeder protocol that receiving a random number in a certain context could be interpreted as implying receipt of a session key and even of statements about the session key. In fact the received message need have no explicit connection to the implicit message at all. The full set of *received messages* for a principal P_i at a point (r,t) includes the explicitly received messages plus any such *implicitly received messages*. While anything might be implied by a message, the implicitly received messages for P_i at (r,t) are limited to the seen messages for P_i at (r,t). Similarly, our model is restricted to runs where principals can only send what they see. Thus, if $\mathit{send}(X,G)$ is in the local history at $r_i(t)$, then X is in the seen messages at $r_i(t)$.

The *said messages* are also a subset of the seen messages; we cannot hold a principal responsible for saying everything that is derivable by him from things he said. For example, if A sends $\{T_a,K,C\}_{K_{ab}}$ to B, we should infer that A said (T_a,K,C). However, even though we can infer that A sees C *sees* K from this action, we should not infer that A said C *sees* K based on it. Given a message M that P_i sends at (r,t), we define the *said submessages* of M by recursively adding to $\{M\}$ the following: (1) the concatenates of all concatenated submessages of M, (2) the unencrypted message of any encrypted submessage of M for which P_i has the encryption key and for which he sees the unencrypted message, (3) the unsigned message in any signed submessage of M for which P_i has the signature key and sees the unsigned message, and (4) the unhashed message in any hashed submessage of M for which he sees the unhashed message. (5) any message M' such that P_i sees M' and P_i meant to imply M' by saying a submessage of M. Implicit in saying that P_i has the key or hash function in the above is that P_i also possesses an algorithm that is feasibly computable in practice by him and that produces the relevant transformation. The set of *said messages* for P_i at (r,t) is the union of the sets

[7] The set of seen messages, and the sets of received and said messages to be defined presently, will be slightly expanded below. Cf. the discussion under **Believing** in §3.2.

of said submessages of all messages that P has sent in r through time t.

3.2 Truth Conditions

We now set out the conditions under which a formula is assigned to be true. We begin by fixing a system, i.e. a set of runs, $\mathcal{R}$. Truth of a formula φ at a point (r,t), written '$(r,t) \models \varphi$', is inductively defined below. '$\models \varphi$' means that φ is valid (true at all points).

Logical Connectives

$(r,t) \models \varphi \wedge \psi$ iff $(r,t) \models \varphi$ and $(r,t) \models \psi$

$(r,t) \models \neg\varphi$ iff $(r,t) \not\models \varphi$[8]

Receiving

$$(r,t) \models P \ \mathit{received}\ X$$

iff X is in the set of received messages for P at (r,t), as defined in §3.1.

Seeing

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

iff X is in the set of seen messages for P at (r,t), as defined in §3.1.

Saying

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

iff, for some message M, at some time $t' \leq t$ in r, P sent M and X is a said submessage of M for P at (r,t'). This gives the truth conditions for P having said X at some point in the past. We also characterize what in means for P to have said X in the current epoch (typically taken to mean since the initial point of the current protocol run).

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

iff, for some message M, at some time $0 \leq t' \leq t$ in r, P sent M and X is a said submessage of M for P at (r,t').

Jurisdiction

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

iff $(r,t) \models P \ \mathit{says}\ \varphi$ implies $(r,t') \models \varphi$ for all $t' \geq 0$. Note that jurisdiction constitutes authority at all points in the current epoch, not just at the time P says φ. This makes it a very strong property. Attributions of jurisdiction are typically part of initial assumptions and should be made sparingly and judiciously.

Freshness A message is fresh if it has not been part of a message sent prior to the current epoch. It is sufficient but not necessary for freshness that a message be unseen prior to the current epoch. A principal might generate a message earlier and not send it until the epoch begins. Truth conditions are thus in terms of the what has been said rather than what has been seen.

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

iff, for all principals P and all times $t' < 0$, $(r,t') \not\models P \ \mathit{said}\ X$.

Keys We will give truth conditions with respect to four types of keys: shared keys, public ciphering keys, public signature keys, and public key-agreement keys. Truth conditions for a shared key to be good for communication between P and Q is a variant of that in [AT91]:

$$(r,t) \models P \stackrel{K}{\leftrightarrow} Q$$

iff, for all t', $(r,t') \models R \ \mathit{said}\ \{X^Q\}_K$ implies either $(r,t') \models R \ \mathit{received}\ \{X^Q\}_K$, or $R = Q$ and $(r,t') \models R \ \mathit{said}\ X$ and $(r,t') \models R \ \mathit{sees}\ K$. If $(r,t') \models R \ \mathit{said}\ \{X\}_K$ (instead of the stronger $(r,t') \models R \ \mathit{said}\ \{X^Q\}_K$), then $R \in \{P,Q\}$ (instead of the stronger $R = P$).

'$\mathrm{PK}(P,K)$' means both that K is a public key associated with principal P *and* that the corresponding private key, K^{-1}, is good. (We refer here to all three types of public keys.) The truth conditions below are thus for both good public key binding and private key secrecy. (We do not mean to imply each principal has only one of any type of public key; however, our notation does assume a unique private key associated with any public key.) Signing and ciphering (encryption) may be separated in the case of public keys. Thus, the two sets of truth conditions for these two types of public keys separate out those features from the shared key truth conditions. The first truth conditions for public keys are for signature keys. Because a principal may come to have beliefs based on a signed message that he cannot produce himself, we need a way to refer to the result of verifying the origin of that message.

$$(r,t) \models \mathrm{SV}(Y,K,X)$$

iff there exists a $\widetilde{K}$ such that it can be verified using K that $Y = [X]_{\widetilde{K}}$.

Note that the truth conditions for $\mathrm{SV}(X,K,Y)$ are not contextual. They hold at one point iff they hold at all points. Thus, we are implicitly assuming that the relevant signature verification algorithm is in A_i for all principals P_i at all points. With this in place we can give the truth conditions associated with public key signature keys.

$$(r,t) \models \mathrm{PK}_\sigma(P,K)$$

iff, and all t', $(r,t') \models Q \ \mathit{received}\ Y \wedge \mathrm{SV}(Y,K,X))$ implies $(r,t') \models P \ \mathit{said}\ X$.

Next we give truth conditions for public ciphering keys.

$$(r,t) \models \mathrm{PK}_\psi(P,K)$$

iff, for all t', $(r,t') \models Q \ \mathit{sees}\ \{X\}_K$ implies $(r,t') \models Q \ \mathit{sees}\ X$ only when $Q = P$.

[8] '$(r,t) \not\models \varphi$' means it is not the case that $(r,t) \models \varphi$.

Truth conditions for key-agreement keys are a bit more complicated:

$$(r,t) \models \mathrm{PK}_{\delta}(P,K)$$

iff for all t',

(1) for some Q and K_q, $(r,t') \models P \overset{F_0(K,K_q)}{\longleftrightarrow} Q$; and,

(2) for all R, K_r, if $(r,t') \not\models R \overset{F_0(K_r,K)}{\longleftrightarrow} P$, then, for all U, K_u, $(r,t') \not\models R \overset{F_0(K_r,K_u)}{\longleftrightarrow} U$. (Here F_0 refers to some agreement function such as that in Diffie-Hellman key agreement that takes the key referred to by the first argument of F_0 and the private cognate of the second argument as its arguments. The first clause guarantees that there is someone with whom P (using K) can form a good key. The second clause guarantees that anyone with whom P using K cannot form a good key cannot form a good key with anybody (at least not using that public key). The truth conditions for $\mathrm{PK}_{\delta}(P,K)$ may seem overly complex. But, we cannot simply require that a session key P produces via agreement with the public key of any Q is good. This is because, even if K were still secret, any given Q's private key-agreement key may have been compromised, compromising $F_0(K,K_q)$. On the other hand, we cannot simply require that if P cannot produce a good session key by agreement with Q, then Q has a bad private key-agreement key. That would lead us into a circularity in determining whether truth conditions are satisfied. The above characterization achieves what is needed while avoiding circularity.

These truth conditions are admittedly complex. One might try to decompose the logic into elements with simpler semantics. However, key agreement is complicated stuff. What our current logic and semantics allow us to do is ignore much of that complexity in our syntactic analysis while knowing that what we have is nonetheless sound. Decomposition would just lead us into algorithm or protocol specific details that should be avoided on the logical level.

Believing Our characterization of belief is based on possible worlds. This approach to characterizing belief was first given by Hintikka in [Hin62]. Since the early eighties it has been applied to distributed computing (one example of such application being that in [AT91]). The idea is that a principal's beliefs in a given state are determined by which worlds (global states) are considered to be possibly the state he is in. From his perspective these worlds are indiscernible from one another. For each principal P_i we can thus define a relation $\sim_i$ that indicates for each world (r,t) which worlds are possible in this manner for P_i. Not surprisingly, this is closely tied to the messsages that are comprehended by P_i at each world, those that he can discriminate to be what they are.

The messages that a principal can comprehend are those that he can ultimately tie back to cleartext he has seen and those that he can relate to previously seen messages. The local state for a principal includes a set of seen messages; however, some of these he will see without comprehension. For example, if he sees a hash $H(X)$ but not X, then he does not comprehend what he's seeing to be $H(X)$. Similarly, if he sees $\{X\}_K$, but does not have the relevant decryption key, then he does not comprehend what he is seeing even if X is available plaintext. Nonetheless, we account for the possibility of, e.g., a principal recognizing that a received message is the encryption with his public key of a message he had previously forwarded without comprehending.

We will now define the *comprehension* of principal P_i in a run r. Note that while principals necessarily decompose received messages top down, it will facilitate understanding if comprehension of messages is set out in a bottom up manner. Since public keys are assumed to be generally available and since principals can therefore verify the structure of messages signed by others even if they cannot form those messages, we must somehow account for this. We therefore define a set A_i^+ to be A_i together with the formation of messages that P_i can verify (such as signatures by other principals). Henceforth '$Cl_i(\alpha)$' refers to the closure of the set α under the rules in A_i^+. Let $comp_i(r,0)$ consist of the closure under A_i^+, of all plaintext that P_i has at the start of the protocol in run r. We assume that each principal P_i receives at most one message at a time. If P_i receives no messages at time t in r, then $comp_i(r,t) = comp_i(r,t-1)$.

Suppose that P_i receives M at (r,t). Let α be the set of all hereditary submessages of M that P_i can form or verify at (r,t). In other words α includes the (immediate) submessages of M, the submessages of submessages, and so on, down to the atomic terms contained in M that are contained in $Cl_i(\{M\} \cup comp_i(r,t-1))$. Some of the members of α will not be understood by P_i. We will now proceed through an iterative construction that will replace any $X \in \alpha$ that is not understood by P_i with $*_x$.

Consider all the $X \in \alpha$ that are atomic ($X \in \mathcal{T}$). If $X \in comp_i(r,t-1)$, then let $X \in \beta_0$. If $X \notin comp_i(r,t-1)$, then let $*_x \in \beta_0$. Also, let $comp_i(r,t-1) \subseteq \beta_0$. Let α_0 be the result of substituting $*_x$ for X in any submessage of a member of α if $*_x \in \beta_0$.

Now, consider all the $X \in \alpha_0$ such that X is the result of a single message formation rule (as given in §1.1) and where the components of the X are members of β_0. If P_i can form or verify X with A_i^+ using those components, then let $X \in \beta_1$. If P_i cannot form or verify X with A_i^+ using those components, then let $*_x \in \beta_1$. Also, let $\beta_0 \subseteq \beta_1$. Let α_1 be the result of substituting $*_x$ for X in any submessage of a member of α_0 if $*_x \in \beta_1$.

Consider all the $X \in \alpha_1$ such that X is the result of a single message formation rule (as given in §1.1) and where the components of the X are members of α_1. If P_i

can form or verify X with A_i^+ using those components, then let $X \in \beta_2$. If P_i cannot form or verify X with A_i^+ using those components, then let $*_x \in \beta_2$. Also, let $\beta_1 \subseteq \beta_2$. Let α_2 be the result of substituting $*_x$ for X in any submessage of a member of α_1 if $*_x \in \beta_2$.

Continuing in this way, we will eventually arrive at a stage n for which the only $X \in \alpha_{n-1}$ under consideration is M itself, with asterisks substituted for appropriate submessages. Either this message is replaced by an asterisk expression at stage n or $\alpha_n = \alpha_{n-1}$. In either case, this is the terminating stage for the construction.

We can then define

$$comp_i(r,t) = Cl_i\alpha_n$$

Note that this construction determines what is comprehended not just for hereditary submessages of a message M just received but also previous messages. For example, suppose P received $\{X\}_K$ at (r,t) but only acquired K at some point (r,t') where $t < t'$. $\{X\}_K$ would be replaced by an asterisk expression in $comp_P(r,t)$. But, assuming X were comprehended, $\{X\}_K$ would appear in $comp_P(r,t')$.

We can use this construction to form a *local message* $M_i(r,t)$ for any message M and any principal P_i and point (r,t). Note that in this construction each submessage of M, including M itself, occurs in α. And, each α_j contains a unique element corresponding to each element of α. Thus, given any message M (received or not, seen or not) we can construct the local message $M_i(r,t)$ for P_i at (r,t) by following the above construction to form the relevant substitutions for subexpressions of M until we construct α_n. $M_i(r,t)$ is simply the element of α_n corresponding to M in α. This construction is only relevant to $comp_i(r,t)$ when M is a message newly received or generated by P_i.

We now expand the sets of seen, received, and said messages to include the locally understood messages. Henceforth, if M is in the set of seen (said, received) messages for P_i at (r,t), then so is $M_i(r,t)$.

For any given run r and principal P_i we now define the *locally comprehended run* r_i^* to be the same as r_i except that wherever any message M occurs in $r_i(t)$, for any t, $M_i(r,t)$ replaces M.

The possibility relation $\sim_i$ for a principal P_i in state (r,t) is defined by

$$(r,t) \sim_i (r',t')$$

iff, $r_i^*(t)$ and $r_i'^*(t')$ can be produced one from the other by a uniform substitution of subscripts on asterisks. For example, $r_i^*(t)$ might be the same as $r_i'^*(t')$ except that $*_j$ occurs in the former everywhere that $*_k$ occurs in the latter, and vice versa.

We can now give truth conditions for belief formulae:

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

iff $(r',t') \models \varphi_i(r',t')$ for all (r',t') such that $(r,t) \sim_i (r',t')$, and $\varphi = \varphi_i(r',t')$ for some such (r',t').

In the sequel we may occasionally write '$\sim_p$' and '$comp_p$' respectively for the possibility relation and comprehension of principal P. Similarly we may write '$M_p(r,t)$' to represent the local message corresponding to M for P at (r,t).

It is obvious that $\sim_i$ is an equivalence relation. By a standard result of modal logic this means that all of the axioms of the system **S5** are valid in this semantics [Che80, Gol92]. Readers familiar with the use of logics of knowledge and belief will recognize this as the most standard logic for representing knowledge. And, such readers may therefore wonder why we have chosen to take this as a logic of belief and why we have included only two of the axioms of **S5** in our axiom set. We see no need to include the other axioms for the applications we envision. It is a simple matter to add them should it be necessary. The subtleties of intuitions regarding knowledge and belief in the context of protocol analysis have been discussed elsewhere [Syv91, Syv92], and we will not delve into that issue here.

This completes the conditions necessary to assign truth values to all formulae in the logic.

3.3 Soundness

Theorem 3.1 SVO is sound: if $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.

In words, the theorem says that, for a set of formulae Γ and a formula φ, if φ is derivable from Γ, then φ is true at any world making all of Γ true. Thus, in a typical protocol analysis, if Γ refers to the premise set, as described in §2, then the effect of this theorem is that if all our assumptions (Γ) are true, then so is any protocol goal (φ) proved from those assumptions. (The truth of the assumptions must be evaluated by means outside the logic, e.g., by evaluating their status in the model of computation via the semantics.)

Proof: This is a typical tedious soundness proof [Che80]: show that the axioms are valid (true at all worlds) and that derivation preserves truth. Proof of validity for all axioms is direct by inspection of the truth conditions given in §3.2. We fill in details for those axioms where the result is neither immediate nor standard. Note that all the axioms for which the validity proof is spelled out below are conditionals. By the truth conditions for '$\supset$', it therefore suffices to show in each case that the consequent of the conditional is true at any world at which the antecedent is true.

Ax1–Ax2. As noted above $\sim_i$ is an equivalence relation, and axioms Ax1 and Ax2 are thus valid by a standard result of modal logic [Che80].

Ax3. $(P \overset{K}{\leftrightarrow} Q \wedge R \textit{ received } \{X^Q\}_K) \supset$
$(Q \textit{ said } X \wedge Q \textit{ sees } K)$

Suppose that $(r,t) \models (P \overset{K}{\leftrightarrow} Q \wedge R \textit{ received } \{X^Q\}_K)$. By the definition of a run, there is a $t' < t$ such that $(r,t') \models R' \textit{ said } \{X^Q\}_K$ for some R'. Then, by the truth conditions for $P \overset{K}{\leftrightarrow} Q$, either $(r,t') \models R' \textit{ received } \{X^Q\}_K$ or $(r,t') \models R' \textit{ said } X$, $(r,t') \models R' \textit{ sees } K$, and $R' = Q$. In our model of computation each run is assumed to have an initial state. Thus, each sent message must be sent a first time (without being previously received). So, there exists a $t'' < t$ and R'' such that $(r,t'') \models R'' \textit{ said } \{X^Q\}_K$ and $(r,t'') \not\models R'' \textit{ received } \{X\}_K$. So, $(r,t'') \models R'' \textit{ said } X$, $(r,t) \models Q \textit{ sees } K$, and $R'' = Q$.

Ax4. $(\mathrm{PK}_\sigma(Q,K) \wedge R \textit{ received } X \wedge \mathrm{SV}(X,K,Y)) \supset$
$Q \textit{ said } Y$

This is immediate from the truth conditions for $\mathrm{PK}_\sigma(Q,K)$, and $SV(X,K,Y)$.

Ax5. $((\mathrm{PK}_\delta(P,K_p)) \wedge (\mathrm{PK}_\delta(Q,K_q))) \supset P \overset{F_0(K_p,K_q)}{\longleftrightarrow} Q$

Suppose that $(r,t) \models (\mathrm{PK}_\delta(P,K_p) \wedge \mathrm{PK}_\delta(Q,K_q))$ but that $(r,t) \not\models P \overset{F_0(K_p,K_q)}{\longleftrightarrow} Q$. Thus, P using K_p cannot form a good shared key with Q using K_q. By clause (2) of the truth conditions for key agreement, if $\mathrm{PK}_\delta(P,K_p)$, this would mean that, for all times t' at (r,t'), Q cannot make a good session key with anyone using K_q. But, this contradicts our initial assumption that $(r,t) \models \mathrm{PK}_\delta(Q,K_q))$.

Ax6. $\varphi \equiv \varphi[F_0(K,K')/F_0(K',K)]$

This is immediate from the definition of F_0.

Ax7–Ax12. The validity of axioms Ax7–Ax12 is immediate from the definitions of received and seen messages.

Ax13. $P \textit{ believes } (P \textit{ sees } F(X)) \supset P \textit{ believes } P \textit{ sees } X$

(where F is any effectively one-one function such that either F or $\widetilde{F}$ is computable in practice by P). Suppose that $(r,t) \models P \textit{ believes } (P \textit{ sees } F(X))$. Let (r',t') be such that $(r,t) \sim_p (r',t')$. Then, $(r',t') \models (P \textit{ sees } F(X))_p(r',t')$ by the truth conditions for belief. Since principal names are assumed to be generally known, and since, by definition, 'F' denotes a function in A_p^+, $(P \textit{ sees } F(X))_p(r',t') = (P \textit{ sees } F(X_p(r',t')))$. Thus, $(r',t') \models (P \textit{ sees } F(X_p(r',t')))$. So, by definition of the seen messages and since $F \in A_p^+$, this is true iff $(r',t') \models (P \textit{ sees } X_p(r',t'))$.

Again, by the truth conditions for belief, for some (r',t') such that $(r,t) \sim_p (r',t')$, $P \textit{ sees } F(X)$ is $(P \textit{ sees } F(X))_p(r',t')$. And, by the above argument, $(P \textit{ sees } F(X))_p(r',t') = (P \textit{ sees } F(X_p(r',t')))$. So, $X = X_p(r',t')$, and $P \textit{ sees } X = P \textit{ sees } X_p(r',t')$. This completes the truth conditions for belief, so $(r,t) \models P \textit{ believes } (P \textit{ sees } X)$.

Ax14–Ax20. The validity of these axioms are all immediate from the relevant truth conditions.

Note that axiom Ax18 says that a function of fresh arguments is itself fresh, *provided* that the function genuinely depends on the fresh argument. Without this provision the axiom is not valid. To see this note that $X = X + 0 \cdot Y$. So, if $P \textit{ said } X$ before the current epoch, without the provision the freshness of Y allows us to conclude that $P \textit{ says } X$. (We refer here to the values of X and $X + 0 \cdot Y$. Obviously the character string 'X' does not equal the character string '$X + 0 \cdot Y$', which does depend on 'Y' to be produced.)

All that remains to be shown for soundness is that all the ways that φ can be derived from Γ preserve truth. There are three cases. (1) If φ is a theorem or member of Γ, then $\Gamma \models \varphi$ trivially. (2) If φ is obtained by modus ponens, then it occurs in a derivation from Γ in which some ψ and $\psi \supset \varphi$ occur earlier. Then by induction on the structure of the derivation and definition of truth conditions, $\Gamma \models \varphi$. (3) Also by a trivial induction, if φ is obtained by necessitation, then φ is $P \textit{ believes } \psi$ for some P and some ψ such that $\vdash \psi$. By inductive hypothesis, $\models \psi$. So, by the truth conditions for belief, $\models P \textit{ believes } \psi$. Thus, a fortiori, $\Gamma \models P \textit{ believes } \psi$. □

4 More Sample Applications

In this section we look at two key agreement protocols. These protocols are often subtler in many ways than standard key distribution protocols. Thus, while these analyses are commensurately subtler than those of, e.g., [BAN89], they also illustrate the relative strength of SVO. Some expressions from VO are useful in these analyses. Whenever notation from VO is encountered it should be read as a syntactic abbreviation as defined from SVO primitives in appendix B.1.

Before beginning analysis of the protocols themselves we set out some generic formal goals that any authentication protocol might be intended to meet. Similarly, we set out some generic assumptions. In our analysis, we prove that each of the protocols meets some of the generic goals presented.

4.1 Generic Formal Goals and Assumptions

We list first some generic goals that protocols to be discussed below will be shown to meet. This is not meant to be taken as a definitive list of *the* goals that a key agreement or key distribution protocol should meet.

A Computations	messages sent	**B** Computations
generate $\overline{x}$, $\overline{R}_a = \alpha^{\overline{x}}$		generate $\overline{y}$, $\overline{R}_b = \alpha^{\overline{y}}$
$\text{Cert}_a = (\overline{R}_a, ID_a, s_t\{\overline{R}_a, ID_a\})$		$\text{Cert}_b = (\overline{R}_b, ID_b, s_t\{\overline{R}_b, ID_b\})$
generate x, $R_a = \alpha^x$	$\longrightarrow \text{Cert}_a, R_a$	generate y, $R_b = \alpha^y$
verify Cert_b; $K = (\overline{R}_b)^x \cdot (R_b)^{\overline{x}}$	$\text{Cert}_b, R_b \longleftarrow$	verify Cert_a; $K = (\overline{R}_a)^y \cdot (R_a)^{\overline{y}}$

Figure 1: The MTI Protocol A(0)

G1. Far-end operative: A *believes* B *says* X

G2. Entity authentication:
A *believes* B *says* $F(X, N_a)$

G3. Secure key establishment: [9]
A *believes* $A \stackrel{K-}{\longleftrightarrow} B$

G4. Key confirmation:
A *believes* $A \stackrel{K+}{\longleftrightarrow} B$

G5. Key freshness: A *believes fresh*(K)

G6. Mutual understanding of shared key:
A *believes* B *says* $B \stackrel{K-}{\longleftrightarrow} A$

The intuitive meaning and reasons for each of these should be clear for the most part. G1 says that A believes B has been online during the current epoch. In G2, N_a is A's nonce, and F is assumed to be an effectively one-one function such that F is computable in practice by B and F or $\widetilde{F}$ is computable in practice by A. The idea is that A is assured that B has recently offered the response 'X' to A's challenge of N_a. (No other understanding of 'entity authentication' is intended.) Note that it is still possible for G3 to hold if B has not participated in the protocol and even if B does not possess session key K.

We now collect some generic formal assumptions, some of which will be made in the analysis of the protocols considered below. They are stated for a principal A and a trusted authority T. In a protocol involving two principals A and B, they may be assumed for either or both principals.

A1. T's signature key: A *believes* $\text{PK}_\sigma(T, K_t)$

A2. T's signature key jurisdiction:
A *believes* T *controls* $\text{PK}_\sigma(B, K_b)$

A3. T's agreement key jurisdiction:
A *believes* T *controls* $\text{PK}_\delta(B, K_b)$

A4. Own agreement key quality:
A *believes* $\text{PK}_\delta(A, K_a)$

A5. Nonce freshness: A *believes fresh*(N_a)

The meaning of all these assumptions should be self evident: principals believe they have good signature keys for trusted authorities, that trusted authorities have jurisdiction over statements concerning the public keys of other principals, that their own agreement keys are good, and that nonces they generate themselves are fresh. As noted in §2.2.1, jurisdiction assumptions are rather strong and should be made with caution. When issuing a certificate, it is generally important that the relevant authority confirm not only the authenticity of the request but also that the requesting principal possesses the corresponding private key. If this were not true for signature or key agreement certificates, then the relevant juridiction assumption would not be true either. The significance of this will become apparent presently. This is not meant to be a comprehensive list of assumptions for any type of protocol.

4.2 The MTI Protocol A(0)

The key agreement protocol A(0) of Matsumoto, Takashima, and Imai [MTI86] results in the establishment of a shared secret key; two Diffie-Hellman exponentiations are used, combining fixed and (per-run) variant parameters, allowing the creation of a unique key for each protocol run while reusing certified public key-agreement keys. A publicly known appropriate prime p and primitive element α in $GF(p)$ are fixed. The parties A and B and the trusted authority T use a common signature scheme in association with certificates; $s_U\{\bullet\}$ denotes the signature of party U. In a preliminary, one-time process, A selects a secret random number $\overline{x}$, computes $\overline{R}_a = \alpha^{\overline{x}}$, and gives this to T; T verifies A's identity and returns a certificate Cert_a consisting of $\overline{R}_a$, a distinguishing identifier ID_a for A, and T's signature over their concatenation. $\overline{R}_a$ serves as A's fixed public key-agreement key, which can now be made available to others by certificate. Similarly, B obtains a secret number y, computes $\overline{R}_b = \alpha^y$, and obtains Cert_b. The protocol between A and B then consists of a single message in each direction, as outlined below and as summarized in Figure 1:

[9] As mentioned above, notation from VO is defined from SVO primitives in appendix B.1.

1. A generates a random positive integer x, computes $R_a = \alpha^x$ and sends R_a to B along with Cert_a.
2. B generates a random positive integer y, computes $R_b = \alpha^y$ and sends R_b to A along with Cert_b.
3. A and B establish the authenticity of each other's certificates by verifying the signature of T thereon using T's known public key, and establish a common key K by respectively computing $K = (\overline{R}_b)^x \cdot (R_b)^{\overline{x}}$ and $K = (\overline{R}_a)^y \cdot (R_a)^{\overline{y}}$.

This protocol is also discussed in [Yac90], where calculations are with respect to an RSA modulus n rather than modulo p as above. Another very similar protocol was given in [Gos90].

4.2.1 Analysis of A(0) protocol

We first specify the protocol in our notation:

$$A \longrightarrow B: \quad (A, \overline{R}_a, [A, \overline{R}_a]_{K_t^{-1}}), R_a$$
$$B \longrightarrow A: \quad (B, \overline{R}_b, [B, \overline{R}_b]_{K_t^{-1}}), R_b$$

We next turn to the formation of the set of premises to be used in formal derivations of protocol goals. The generic assumptions we make correspond to **A1**, **A3**, **A4**, and **A5** above. Specifically we assume that each principal A and B believes that K_t is the signature verification key for the trusted authority, T (**A1**), that each principal believes his own agreement key is good (**A4**), and that each principal believes that the key parameters he generates for the protocol are fresh (**A5**). We assume that principals each accept the jurisdiction of T over the agreement key of the other; however, **A3** is not adequate to express this assumption for two reasons. First, by virtue of the semantics for *controls* , it grants jurisdiction only to statements made by T during the current epoch. This protocol relies on statements made by T with no freshness indicators included. Second, the session key is formed by two public pieces of data from principals, but the statement from the trusted authority only concerns one of these. Fortunately, for the purposes of this protocol analysis we can replace **A3** with the more specific assumptions that

$$A \textit{ believes } ((T \textit{ said } PK_\delta(B, \overline{R}_b) \wedge A \textit{ received } ((B, \overline{R}_b, [(B, \overline{R}_b)]_{K_t^{-1}}), R_b)) \supset PK_\delta(B, (\overline{R}_b, R_b)))$$

and similarly for B. We include all these initial assumptions in the premise set. Other initial assumptions reflected in the premise set are that each principal comprehends his own agreement key components and that each principal correctly assesses the result of the verifying T's signature on the other's certificate. The premise set also reflects the messages that each principal receives. Also, recall that any premise set reflects A's comprehension of messages received by including A *believes* A *received* X for each message X that A is assumed to comprehend (and similarly for B). Finally, the set includes premises reflecting the receiver's interpretation of message content for each received message. We now enumerate the premise set.

P1 $A \textit{ believes } PK_\sigma(T, K_t)$
$B \textit{ believes } PK_\sigma(T, K_t)$

P2 $A \textit{ believes } A \textit{ sees } (\overline{R}_a, R_a, \overline{x}, x)$
$B \textit{ believes } B \textit{ sees } (\overline{R}_b, R_b, \overline{y}, y)$

P3 $A \textit{ believes } SV([(B, \overline{R}_b)]_{K_t^{-1}}, K_t, (B, \overline{R}_b))$
$B \textit{ believes } SV([(A, \overline{R}_a)]_{K_t^{-1}}, K_t, (A, \overline{R}_a))$

P4 $A \textit{ believes } ((T \textit{ said } PK_\delta(B, \overline{R}_b) \wedge A \textit{ received } ((B, \overline{R}_b, [(B, \overline{R}_b)]_{K_t^{-1}}), *_b)) \supset (PK_\delta(B, (\overline{R}_b, *_b))))$

$B \textit{ believes } ((T \textit{ said } PK_\delta(A, \overline{R}_a) \wedge B \textit{ received } ((A, \overline{R}_a, [(A, \overline{R}_a)]_{K_t^{-1}}), *_a)) \supset (PK_\delta(A, (\overline{R}_a, *_a))))$

P5 $A \textit{ believes } PK_\delta(A, (\overline{R}_a, R_a))$
$B \textit{ believes } PK_\delta(B, (\overline{R}_b, R_b))$

P6 $A \textit{ believes } \textit{fresh}(R_a)$
$B \textit{ believes } \textit{fresh}(R_b)$

P7 $A \textit{ received } ((B, \overline{R}_b, [B, \overline{R}_b]_{K_t^{-1}}), R_b)$
$B \textit{ received } ((A, \overline{R}_a, [A, \overline{R}_a]_{K_t^{-1}}), R_a)$

P8 $A \textit{ believes } A \textit{ received } ((B, \overline{R}_b, [B, \overline{R}_b]_{K_t^{-1}}), *_b)$
$B \textit{ believes } B \textit{ received } ((A, \overline{R}_a, [A, \overline{R}_a]_{K_t^{-1}}), *_a)$

P9 $A \textit{ believes } (T \textit{ said } (B, \overline{R}_b) \supset T \textit{ said } PK_\delta(B, \overline{R}_b))$
$B \textit{ believes } (T \textit{ said } (A, \overline{R}_a) \supset T \textit{ said } PK_\delta(A, \overline{R}_a))$

We now turn to formal derivations. In the interest of brevity, we will compress many of the steps together, and we will not cite the use of propositional reasoning in giving the justifications for derivation lines. Since there is nothing in the protocol to authenticate either principal to the other in any way, there is no hope of deriving the generic goals **G1**, **G2**, **G4**, or **G6**. We give formal derivations of goals **G3** and **G5** beginning with **G3** ($A \textit{ believes } A \overset{K-}{\longleftrightarrow} B$).

1. $A \textit{ believes } A \textit{ received } ([PK_\delta(B, \overline{R}_b)]_{K_t^{-1}})$
   by P8, Ax1, Ax7, Nec, MP
2. $A \textit{ believes } T \textit{ said } (B, \overline{R}_b)$
   by 1, P1, P3, Ax1, Ax4, Nec, MP

3. A *believes* T *said* $PK_\delta(B, \overline{R}_b)$
   by 2, P9, Ax1, MP
4. A *believes* $PK_\delta(B, (\overline{R}_b, *_b))$
   by 2, P4, Ax1, MP
5. A *believes* $PK_\delta(A, (\overline{R}_a, R_a))$
   by P5
6. A *believes* $(A \overset{K}{\leftrightarrow} B)$
   by 4, 5, Ax1, Ax5, Nec, MP,
   where $K = F_0((\overline{R}_a, R_a), (\overline{R}_b, R_b))$
7. A *believes* A *sees* $(\overline{R}_b, *_b)$
   by P8, Ax1, Ax10, Ax11, Ax12, Nec, MP
8. A *believes* A *sees* K
   by 7, P2, Ax11, Ax12, Ax1, Nec, MP
   where $K = F_0((\overline{R}_a, R_a), (\overline{R}_b, R_b))$
9. A *believes* $(A \overset{K-}{\longleftrightarrow} B)$
   by 6, 8, Ax1, MP, and def. of $A \overset{K-}{\longleftrightarrow} B$.

The derivation of **G3** for B is virtually identical.

As Burrows et al. found in their analyses in [BAN89], it is often instructive to look at the assumptions necessary to derive a goal. We have noted before that jurisdiction assumptions are powerful and should be made judiciously. We thus delve more deeply into premise **P4**. First note that the quality and binding of the entire agreement key is assumed based only on the trusted authority's assertion concerning the long term part (and the comprehensibility of the fresh part). This is an unavoidable assumption since the fresh part of each public agreement key is sent only in the clear. If this cleartext were attacked it could result in principals believing that they share a good session key. This attack in no way invalidates the above result since A does have K, and K is a session key good for at most A and B (though in actuality good for nobody, if the attacker tampers as indicated above).

Another assumption implicit in **P4** is, however, more serious. Specifically, **P4** (and more generally **A3**) assumes that the trusted authority has jurisdiction over the binding and quality of a principal's agreement key. This is the danger we alluded to above if the trusted authority issues a certificate without checking both that the certificate matches an authenticated request and that the requesting principal has the corresponding private key. In this protocol, should T issue certificates without this confirmation, it would be possible for a principal E to trick another principal B into thinking he has formed a session key with E when B has in fact formed a session key with A. In this case the above result would be spurious. Here is an account of the attack. (A slightly more complicated attack having similar results is given in [MQV95].)

Attack on the A(0) Protocol

1. E obtains $\overline{R}_a$, A's public long term agreement key, perhaps by legitimate sessions of this protocol. E requests and receives a certificate, $\mathrm{Cert}_e = (\overline{R}_a, ID_e, s_t\{\overline{R}_a, ID_e\})$. Note that E does not obtain $\overline{x}$.
2. A initiates a legitimate session with B. That is, A generates a random positive integer x, computes $R_a = \alpha^x$ and sends R_a to B along with certificate Cert_a.
3. E intercepts A's message, substitutes Cert_e for Cert_a and forwards (R_a, Cert_e) to B.
4. B generates a random positive integer y, computes $R_b = \alpha^y$ and sends R_b to E along with certificate Cert_b.
5. E forwards (R_b, Cert_b) to A.
6. A and B establish the authenticity of received certificates by verifying the signature of T thereon using T's known public key, and establish a common key K by respectively computing $K = (\overline{R}_b)^x \cdot (R_b)^{\overline{x}}$ and $K = (\overline{R}_a)^y \cdot (R_a)^{\overline{y}}$. While A correctly believes that K is a session key for communication with B, B erroneously believes that this key is for communication with E.

Next we give a formal derivation of goal **G5**, A *believes fresh*(K).

1. A *believes fresh*(R_a)
   by P6
2. A *believes fresh*(K)
   by 1, Ax18, Ax1, Nec, MP, and def. of K

Note that while we are able to formally derive key freshness, we must implicitly assume that B is competent in his choice of short and long term agreement keys. For example, if he were to choose $\overline{y} \equiv 0 \pmod{p-1}$, then the K would not depend on R_a, and the derivation of freshness would be spurious.

4.3 The STS Protocol

We next review the authenticated key agreement protocol of Diffie, van Oorschot and Wiener called the "Station-to-Station" (STS) protocol [DvOW92]. A publicly known appropriate prime p and primitive element α in $GF(p)$ are fixed for use in Diffie-Hellman key exchange. Parties A and B use a common signature scheme: $s_U\{\bullet\}$ indicates the signature on the specified argument using the private signature key of party U. $E_K(\bullet)$ indicates the symmetric encryption of the specified argument using algorithm E under key K. Public key certificates are used to make the public signature keys of A and B available to each other. In a one-time process prior to the exchange between A and B, each party must present to T his true identity and public key (e.g., ID_a, K_a), have T verify his true identity by some

A Computations	messages sent	**B** Computations
$\text{Cert}_a = (K_a, ID_a, s_t\{K_a, ID_a\})$		$\text{Cert}_b = (K_b, ID_b, s_t\{K_b, ID_b\})$
generate x, $R_a = \alpha^x$	$\longrightarrow R_a$	generate y, $R_b = \alpha^y$; $K = (R_a)^y$
$K = (R_b)^x$; verify $\text{Cert}_b, \text{Token}_{ba}$	$R_b, \text{Cert}_b, \text{Token}_{ba}, \longleftarrow$	$\text{Token}_{ba} = E_K(s_b\{R_b, R_a\})$
$\text{Token}_{ab} = E_K(s_a\{R_a, R_b\})$	$\longrightarrow \text{Cert}_a, \text{Token}_{ab}$	verify $\text{Cert}_a, \text{Token}_{ab}$

Figure 2: The STS Protocol

(typically non-cryptographic) means, and then obtain from T his own certificate. The protocol is as follows.

1. A generates a random positive integer x, computes $R_a = \alpha^x$ and sends R_a to a second party.
2. Upon receiving R_a, B generates a random positive integer y, computes $R_b = \alpha^y$ and $K = (R_a)^y$.
3. B computes the authentication signature $s_b\{R_b, R_a\}$ and sends to A the encrypted signature $\text{Token}_{ba} = E_K(s_b\{R_b, R_a\})$ along with R_b and his certificate Cert_b. Here ',' denotes concatenation.
4. A receives these values and from R_b computes $K = (R_b)^x$.
5. A verifies the validity of B's certificate by verifying the signature thereon using the public signature verification key of the trusted authority. If the certificate is valid, A extracts B's public signature key, K_b from Cert_b.
6. A verifies the authentication signature of B by decrypting Token_{ba}, and using K_b to check that the signature on the decrypted token is valid for the known ordered pair R_b, R_a.
7. A computes $s_a\{R_a, R_b\}$ and sends to B her certificate Cert_a and $\text{Token}_{ab} = E_K(s_a\{R_a, R_b\})$.
8. Analogously, B checks Cert_a. If valid, B extracts A's public signature key K_a and proceeds.
9. Analogously, B verifies the authentication signature of A by decrypting Token_{ab}, and checking the signature on it using K_a and knowledge of the expected pair of data R_a, R_b.

The protocol is successful from each party's point of view if signature verification succeeds on both the received certificate and authentication signature. In this case, the protocol provides assurance that a shared secret has been jointly established with the party identified in the received certificate.

Figure 2 provides a summary of the messages exchanged, and actions taken, by each of the parties in this protocol.

4.3.1 Analysis of STS protocol

The specification of the STS protocol in our notation is as follows:

$$A \longrightarrow B : R_a$$
$$B \longrightarrow A : R_b, (B, K_b, [B, K_b]_{K_t^{-1}}), \{[R_b, R_a]_{K_b^{-1}}\}_K$$
$$A \longrightarrow B : R_a, (A, K_a, [A, K_a]_{K_t^{-1}}), \{[R_a, R_b]_{K_a^{-1}}\}_K$$

We include in the premise set generic assumptions corresponding to **A1** (trusting the authority's signature key), **A2** (trusting the authority's jurisdiction over signature keys), **A4** (trusting the quality one's own agreement key), and **A5** (trusting the freshness of one's own agreement key). As with the MTI protocol A(0), in the STS protocol it is assumed that the trusted authority has timeless jurisdiction over signatures; thus, we cannot use **A2** as stated above. The appropriate variant is trivial to determine and appears as premise **P3** below.

We saw in analyzing the A(0) protocol the subtlety of jurisdiction assumptions. For reasons similar to the ones discussed in connection with the attack on A(0) above we cannot allow principals to have jurisdiction over the quality and binding of their own agreement keys. (That is, we cannot unless other protections are in place, e.g., in the case of STS a signature or keyed hash.) However, some related assumption is necessary if we are to derive any results about the quality of the session key K. Consequently, we record as one of our formal assumptions

$$A \textit{ believes } ((A \textit{ received } \{[*_b, R_a]_{K_b^{-1}}\}_K \wedge PK_\sigma(B, K_b) \wedge PK_\delta(A, R_a)) \supset PK_\delta(B, *_b))$$

the legitimacy of which we now proceed to justify.

First, we may assume that honest principals are competent enough to not encrypt or sign messages blindly, i.e., without any understanding of the message content. So, if A did not recognize R_a within $[*_b, R_a]_{K_b^{-1}}$, then A would not encrypt $[*_b, R_a]_{K_b^{-1}}$ with K. If A did recognize R_a in $[*_b, R_a]_{K_b^{-1}}$, then she may be assumed to be competent to recognize it as only to be used within this protocol and again would not encrypt this with K.

Thus, if A *believes* (A *received* $\{[*_b, R_a]_{K_b^{-1}}\}_K$) then A believes that someone other than herself said it. Given that A *believes* $PK_\delta(A, R_a)$, A can also confirm that $K = *_b{}^x$, hence that $*_b$ is a public agreement key.

We will independently, formally derive below that A believes B's signature key to be good for B. Thus, A can infer that B signed R_a together with either his own agreement key or someone else's. If $*_b$ is his own, then $PK_\delta(B, *_b)$. Assume that $*_b$ is not B's agreement key. Thus, he can only have signed blindly, i.e., without knowing the significance of R_a or $*_b$. But, this violates competency if B is honest. If B is dishonest, then either he has broken the private agreement key corresponding to $*_b$ or the principal corresponding to $*_b$ has been tricked into encrypting $[*_b, R_a]_{K_b^{-1}}$ with K. The first possibility is implicitly assumed not to have occurred. (Similarly, it is assumed that no one other than B has B's private signature key.) And, the second possibility is ruled out by an argument similar to that in the last paragraph. Hence A is justified in inferring that B produced the received message and therefore that $PK_\delta(B, *_b)$. A similar argument justifies the corresponding assumption for B.

We also assume that honest principals are competent to use the public keys they generate for a protocol run only within that run and to properly execute the protocol. In practice this allows us to assume a principal can recognize the message(s) sent by the other principal in the protocol as not having originated with herself. This is reflected in the premise set as **P10**.

Finally, the premise set includes the usual assumptions about what principals received, what they comprehend, and how they interpret received messsages.

We now enumerate the premise set.

P1 A *believes* $PK_\sigma(T, K_t)$
B *believes* $PK_\sigma(T, K_t)$

P2 A *believes* $SV([B, K_b]_{K_t^{-1}}, K_t, (B, K_b))$
A *believes* $SV([*_b, R_a]_{K_b^{-1}}, K_b, (*_b, R_a))$
B *believes* $SV([A, K_a]_{K_t^{-1}}, K_t, (A, K_a))$
B *believes* $SV([*_a, R_b]_{K_a^{-1}}, K_a, (*_a, R_b))$

P3 A *believes* ((T *said* $PK_\sigma(B, K_b)) \supset PK_\sigma(B, K_b)$)
B *believes* ((T *said* $PK_\sigma(A, K_a)) \supset PK_\sigma(A, K_a)$)

P4 A *believes* $PK_\delta(A, R_a)$
B *believes* $PK_\delta(B, R_b)$

P5 A *believes* *fresh*(R_a)
B *believes* *fresh*(R_b)

P6 A *believes* A *sees* (R_a, x)
B *believes* B *sees* (R_b, y)

P7 A *received*
$R_b, (B, K_b, [B, K_b]_{K_t^{-1}}), \{[R_b, R_a]_{K_b^{-1}}\}_K$
B *received*
$R_a, (A, K_a, [A, K_a]_{K_t^{-1}}), \{[R_a, R_b]_{K_a^{-1}}\}_K$

P8 A *believes* A *received*
$*_b, (B, K_b, [B, K_b]_{K_t^{-1}}), \{[*_b, R_a]_{K_b^{-1}}\}_K$
B *believes* B *received*
$*_a, (A, K_a, [A, K_a]_{K_t^{-1}}), \{[*_a, R_b]_{K_a^{-1}}\}_K$

P9 A *believes* ((A *received* $\{[*_b, R_a]_{K_b^{-1}}\}_K \wedge$
$PK_\sigma(B, K_b) \wedge PK_\delta(A, R_a)) \supset$
$PK_\delta(B, *_b)$)
B *believes* ((B *received* $\{[*_a, R_b]_{K_a^{-1}}\}_K \wedge$
$PK_\sigma(A, K_a) \wedge PK_\delta(B, R_b)) \supset$
$PK_\delta(A, *_a)$)

P10 A *believes* $\neg$(A *said* $\{[*_b, R_a]_{K_b^{-1}}\}_K$)
B *believes* $\neg$(B *said* $\{[*_a, R_b]_{K_a^{-1}}\}_K$)

P11
A *believes* (T *said* $(B, K_b) \supset T$ *said* $PK_\sigma(B, K_b)$)
B *believes* (T *said* $(A, K_a) \supset T$ *said* $PK_\sigma(A, K_a)$)

We now derive formal goals **G1**–**G5** for the STS protocol.

1. A *believes* A *received* $[(B, K_b)]_{K_t^{-1}}$
   by P8, Ax1, Ax7, Nec, MP
2. A *believes* T *said* $PK_\sigma(B, K_b)$
   by 1, P1, P2, P11, Ax1, Ax4, Nec, MP
3. A *believes* $PK_\sigma(B, K_b)$
   by 2, P3, Ax1, MP
4. A *believes* A *received* $\{[*_b, R_a]_{K_b^{-1}}\}_K$
   by P8, Ax1, Ax7, Nec, MP
5. A *believes* $PK_\delta(B, *_b)$
   by 3, 4, P4, P9, Ax1, MP
6. A *believes* $A \overset{K}{\leftrightarrow} B$
   by 5, P4, Ax1, Ax5, Nec, MP
   (where $K = F_0(R_a, *_b)$)
7. A *believes* A *sees* K
   by P8, P6, Ax1, Ax10, Ax11, Ax12, Nec, MP
   (where $K = F_0(R_a, *_b)$)
8. A *believes* $A \xleftrightarrow{K-} B$
   by 6, 7, Ax1, MP, and def. of $A \xleftrightarrow{K-} B$
9. A *believes* *fresh*(K)
   by P5, Ax18, MP (where $K = F_0(R_a, *_b)$)
10. A *believes* $*$*confirm*$_A(K)$
    by 4, 9, P10, Ax1, MP,
    and def. of $*$*confirm*$_A(K)$
11. A *believes* $A \xleftrightarrow{K+} B$
    by 8, 10, Ax1, MP, and def. of $A \xleftrightarrow{K+} B$

12. *A believes A received* $[*_b, R_a]_{K_b^{-1}}$
by 4, 7, Ax1, Ax8, Nec, MP

13. *A believes B said* $(*_b, R_a)$
by 3, 12, P2, Ax1, Ax4, Nec, MP

14. *A believes B says* $(*_b, R_a)$
by 13, P5, Ax1, Ax19, Nec, MP

Goal **G1** is a special case of **G2**, which is derived in line 14. **G3** is derived in line 8, **G4** in line 11, and **G5** in line 9. A similar proof shows that all of these goals are formally derivable for B from the same premise set.

There is no possibility of deriving **G6** (mutual understanding of shared key) for A. However, it would be possible to derive **G6** for B with a minimally revised premise set. It is a standard part of BAN idealization to interpret the first message from a principal employing appropriate use of a shared encryption key as including the assertion that the key is good for the relevant principals. Thus, we might add a premise allowing B to interpret receipt of $\{[*_a, R_b]_{K_a^{-1}}\}_K$ as implying receipt of $\{[*_a, R_b, A \stackrel{K}{\leftrightarrow} B]_{K_a^{-1}}\}_K$. This would be sufficient to allow the derivation of **G6** for B. But, as always with such interpretations, we must be very careful. (Recall the earlier discussion regarding problems hidden by assumptions in the idealization of NS.) It would be incorrect to so interpret the message from B to A. By the end of a successful protocol run B believes he has a good key for communication with A; nonetheless, until he receives the last message he has no guarantee that it is A with whom he is establishing a key. He has received nothing from A when he sends his message except a cleartext number that should appear random to him. (Presumably he also has an indicator of who sent the number, but this is not assumed to be protected in any way.) Thus, it would be wrong for A to interpret B's message as including an assertion from him that $B \stackrel{K}{\leftrightarrow} A$. This could only be reasonably stated by B in a further message, subsequent to the last one he receives in this protocol.

In [Low96] Lowe constructs an "attack" on STS. It is an attack because "A believes that B thought that he (B) was talking to A." (p. 165) The above discussion shows that such could not constitute an attack on STS because this was never a goal of the protocol, nor was it stated to be in [DvOW92]. In fact, in [vO93b] it was noted that such an "eager belief" on A's part should be taken as unverified since it assumes B's reception and processing of the third message. But, in the Lowe attack on STS, B does not complete the protocol. A is entitled to infer entity authentication of B (**G2**), and this remains true in the attack Lowe constructs. But, A is not entitled to conclude that she has mutual understanding with B (**G6**) or anything similar.

5 Conclusions and Further Study

In this paper we have presented a logic that encompasses four of its predecessors in the BAN family. We have also presented a model-theoretic semantics for our logic with respect to which it is sound. Despite adding expressiveness and axioms sufficient to reason about all the properties of cryptographic protocols to which these four predecessors are addressed, it is no more syntactically complex than any of them. In fact, measured by the number of rules or axioms and their relative simplicity, it is less complex than GNY, AT, and VO. And, it has about the same number as BAN. In sum, we believe this logic to be about as simple to use as any of those from which it is derived; yet it is more expressive than any of them. Indeed, our analysis of the Needham-Schroeder protocol compares favorably in simplicity to the one in [BAN89]. It also uncovers a previously unnoticed feature of the NS protocol. This led us to more precisely delimit application context assumptions and goals for the protocol than did either the original [NS78] or the analysis in [BAN89].

We have also analyzed two key agreement protocols. The structure of these is rather subtle and analysis commensurately more complex than for simple key distribution protocols of the type analyzed in [BAN89]. Nonetheless, we used the logic to derive a number of desirable goals for the protocols analyzed. And, by taking a closer look at the assumptions necessary to derive those goals, we were lead to find an attack on one of them. We reiterate that one of the virtues of formal protocol analysis is that it forces one to fully set out the formal assumptions necessary for a derivation. And, one of the virtues of a model-theoretic semantics is that it presents a mathematically rigorous setting in which to evaluate the truth of those assumptions.

We have not looked at all the logics that have been derived from BAN, e.g., [MB93]. (That logic is a contraction rather than an expansion of BAN. It is designed to allow much that is informal in the analysis process to be automated.) In particular we have not discussed logics that express either time or message ordering. The goals of these logics are somewhat more ambitious than those discussed above. One of those goals is to address more types of replay attacks. BAN is only directed at classic replays, i.e., replays of messages originally sent before the current protocol began. GNY, with its not-originated-here syntax, adds the ability to reason about some replay attacks using messages from within the current protocol run but still does not address interleaving attacks, that is attacks involving replay of messages from at least two contemporaneous protocol runs. (Cf. [BGH+92], [DvOW92], [Sne92], [Car93].) Indeed, none of the logics discussed in this paper generally addresses interleavings at all. (One might, nonetheless, uncover an interleaving attack by coincidence in the course of analysis using one of these logics. The point is that there are no features of these logics that address such attacks.)

Failure of methods such as BAN logic to address interleaving attacks has led some to focus on the notion of current protocol run rather than on freshness. However, this still leaves some types of replays unaddressed (e.g., the first attack presented in [Syv93b]). We also have not explored the relationship between different BAN-like logics that reason about time (e.g., [GS91]) or the relationship they have to logics that allow reasoning about message ordering (e.g., [KG91]). Our suspicion is that the logics of [GS91] and [KG91] can be captured by the logic of this paper with the temporal additions of [Syv93a].

Finally, we have not looked at the still more ambitious project of unifying the BAN family with other types of logics. Nonetheless, we have produced a unified BAN-like logic that captures the features of four other BAN-like logics. We have approached this from the perspective of having an integrated model. The result is more than a collection of tools. Indeed, we believe it to be a better instance of all the tools it contains.

References

[AT91] Martín Abadi and Mark Tuttle. A Semantics for a Logic of Authentication. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, August 1991.

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. Research Report 39, Digital Systems Research Center, February 1989. Parts and versions of this material have been presented in many places including *ACM Transactions on Computer Systems*, 8(1): 18–36, Feb. 1990. All references herein are to the SRC Research Report 39 as revised Feb. 22, 1990.

[BGH+92] Ray Bird, Inder Gopal, Amir Herzberg, Phil Janson, Shay Kutten, Refik Molva, and Moti Yung. Systematic Design of Two-Party Authentication Protocols. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1992.

[BM92] S.M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks. In *Proc. IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84. IEEE Computer Society Press, Los Alamitos, California, May 1992.

[BM93] S.M. Bellovin and M. Merritt. Augmented Encrypted Key Exchange: a Password-based Protocol Secure Against Dictionary Attacks and Password File Compromise. In *Proc. First ACM Conference on Computer and Communications Security*, pages 244–250, November 1993.

[Car93] Ulf Carlsen. Using Logics to Detect Implementation-Dependent Flaws. In *Proceedings of the Ninth Annual Computer Security Applications Conference*, pages 64–73. IEEE Computer Society Press, Los Alamitos, California, December 1993.

[Che80] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, 1980.

[DH76] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[DS81] D. E. Denning and G. M. Sacco. Timestamps in Key Distribution Protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and Authenticated Key Exchanges. *Designs, Codes, and Cryptography*, 2:107–125, 1992.

[GNY90] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, Los Alamitos, California, 1990.

[Gol92] Robert Goldblatt. *Logics of Time and Computation, 2^{nd} edition*, volume 7 of *CSLI Lecture Notes*. CSLI Publications, Stanford, 1992.

[Gos90] K.C. Goss. Cryptographic Method and Apparatus for Public Key Exchange with Authentication, 1990. U.S. Patent 4,956,863 (granted Sept. 11, 1990).

[GS90] Klaus Gaarder and Einar Snekkenes. On the formal analysis of PKCS authentication protocols. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology — AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 106–121. Springer-Verlag, 1990. These are the proceedings of AUSCRYPT'90, Jan. 8-11, 1990.

[GS91] Klaus Gaarder and Einar Snekkenes. Applying a Formal Analysis Technique to the CCIT X.509 Strong Two-Way Authentication Protocol. *Journal of Cryptology*, 3:81–98, 1991. A preliminary version of this paper appeared as [GS90].

[Hin62] Jaakko Hintikka. *Knowledge and Belief: An Introduction to the Logic of Two Notions.* Cornell University Press, Ithaca, N.Y., 1962.

[KG91] Rajashekar Kailar and Virgil D. Gligor. On Belief Evolution in Authentication Protocols. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 103–116. IEEE Computer Society Press, Los Alamitos, California, 1991.

[Low96] Gavin Lowe. Some New Attacks upon Security Protocols. In *Proceedings of the 9th Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society Press, Los Alamitos, California, 1996.

[MB93] Wenbo Mao and Colin Boyd. Towards a Formal Analysis of Security Protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, pages 147–158. IEEE Computer Society Press, Los Alamitos, California, 1993.

[Men87] Elliott Mendelson. *Introduction to Mathematical Logic.* Wadsworth Publishing Co., 1987.

[MQV95] Alfred Menezes, Minghua Qu, and Scott Vanstone. Some New Key Agreement Protocols Providing Implicit Authentication, 1995. Preprint.

[MTI86] T. Matsumoto, Y. Takashima, and H. Imai. On Seeking Smart Public-Key Distribution Systems. *Trans. IECE Japan*, 69(2):99–106, February 1986.

[Nes90] D. M. Nessett. A Critique of the Burrows, Abadi, and Needham Logic. *Operating Systems Review*, 24(2):35–38, April 1990.

[NS78] R.M. Needham and M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[Sne92] Einar Snekkenes. Roles in Cryptographic Protocols. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Computer Society Press, Los Alamitos, California, 1992.

[SvO94] Paul F. Syverson and Paul C. van Oorschot. On Unifying Some Cryptographic Protocol Logics. In *Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 14–28. IEEE Computer Society Press, Los Alamitos, California, 1994.

[Syv91] Paul F. Syverson. The Use of Logic in the Analysis of Cryptographic Protocols. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 156–170. IEEE Computer Society Press, Los Alamitos, California, 1991. A corrected discussion of many of the issues in this paper appeared in [Syv92].

[Syv92] Paul F. Syverson. Knowledge, Belief, and Semantics in the Analysis of Cryptographic Protocols. *Journal of Computer Security*, 1(3):317–334, 1992.

[Syv93a] Paul F. Syverson. Adding Time to a Logic of Authentication. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 97–101. ACM Press, New York, November 1993.

[Syv93b] Paul F. Syverson. On Key Distribution Protocols for Repeated Authentication. *Operating Systems Review*, 27(4):24–30, October 1993.

[TMN90] Makoto Tatebayashi, Natsume Matsuzaki, and David B. Newman, Jr. Key Distribution Protocol for Digitial Mobile Communication Systems. In G. Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1990.

[vO93a] Paul C. van Oorschot. An Alternative Explanation of two BAN-logic "failures". In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 443–447. Springer-Verlag, 1993.

[vO93b] Paul C. van Oorschot. Extending Cryptographic Logics of Belief to Key Agreement Protocols (Extended Abstract). In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 232–243, November 1993.

[Yac90] Y. Yacobi. A Key Distribution Paradox. In *Advances in Cryptology — CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 268–273. Springer Verlag, Berlin, 1990.

A Relation to GNY extensions

In [GNY90], Gong, Needham, and Yahalom presented GNY. This logic is noteworthy for making one of the largest additions to both the notation and logical rules of BAN. It is therefore interesting to see how much of it is easily accomodated in SVO. This is investigated in this appendix. Similar investigation is made of VO in the next one.

A.1 GNY Notational Additions

$P \triangleleft X$: P is *told* X. This is expressed in our syntax as 'P *received* X'.

$P \ni X$: P *possesses*, or is capable of possessing X. This is expressed in our syntax as 'P *sees* X'.

$P \mid\!\sim X$: P *once conveyed* X. This is expressed in SVO as 'P *said* X'.

$\#(X)$: X is *fresh*. This is expressed in SVO as '*fresh*(X)'.

$\phi(X)$: Recognizability of X. In GNY rules this only occurs in the context of someone's belief. This is consistent with the reasonable requirement that recognizability be tied to an individual, rather than considering what is recognizable to everyone. We will express this relativization in SVO by translating expressions of the form $P \models \phi(X)$ in GNY as P *believes* P *sees* X. This is relativization is discussed below when we look at GNY recognizability rules.

$P \triangleleft *X$: P is told a formula that he did not convey previously in the current run. This is captured in SVO as '$(P$ *received* $X) \wedge \neg(P$ *says* $X)$'. Note that the SVO expression is actually broader than the GNY expression. It says that P did not say X since the start of the current run, whether within the run or not.

$X \rightsquigarrow C$: These are called message extensions. They are used in conveyed messages to indicate conditionality of an assertion. They are only used logically in connection with GNY J2, one of the jursdiction rules. We defer comment to the section below where we discuss this rule.

It is interesting that we were unable to give translations for some of the GNY formulae without referring to the corresponding logical rules. This is because, beyond a minimal intuitive explanation, any technical meaning that GNY expressions hold is tied up with the logic.

A.2 GNY Logical Rules

We will look at these rules with the following question in mind. Once we have made an appropriate translation to SVO syntax, is there a logical derivation (in SVO) of the conclusion of a rule from its premises? If so, then the rule expresses a result that is syntactically captured in SVO. (Hence, we know that it is also semantically captured by our model of computation because of soundness.) When we say that a GNY rule is derivable in SVO below we mean that the answer to the question just asked is yes.

GNY Rationality Rule
This rule says that whenever we can infer $C2$ from $C1$, we can also infer $P \models C2$ from $P \models C1$. It falls out of the modus ponens rule and axiom Ax1.

GNY Being Told and Possession Rules
All of these rules are obviously derivable in SVO except T5. T5 says that $P \triangleleft Y$ follows from $P \triangleleft F(X, Y)$ and $P \ni X$. F is taken to be a many-to-one computationally feasible function that is one-to-one computationally feasible if either X or Y is held constant, as is its inverse. ([GNY90], p. 235.) It is difficult to assess such a rule in general, but Gong et al. do provide one example of the type of function they have in mind, viz: exclusive-or. Our discussion of T5 thus follows their example. If we view exclusive-or as encryption, then T5 can be viewed as a general statement of T3, which says that $P \triangleleft Y$ follows from $P \triangleleft \{Y\}_X$ and $P \ni X$. However, care must be taken in such cases because, when exclusive-or is used for encryption, $\{X\}_Y = \{Y\}_X$. Strictly speaking, in our language this is only true when both X and Y are keys since $\{X\}_Y$ is only well-formed when Y is a key. Nonetheless, according to T5 in GNY, if P receives $X \oplus Y$ and P possesses both X and Y, then, P has been told X and been told Y. There may be applications for which this is a reasonable inference, but the example shows why we might not want to have T5 as a logical rule. Often, if not virtually always, we would like to distinguish a message sent from attendant parameters, such as keys used to encrypt the message. However, T5 obliterates this distinction by treating the arguments of F symmetrically. Furthermore, such symmetry can serve as the basis of attacks that allow a penetrator to deduce keys from chosen, known, or guessed plaintext—for example, the Simmons attack on the TMN protocol discussed in [TMN90]. This example does not serve as a similar basis for criticism of T3. The symmetry in the encryption algorithm subjects it to direct attack. This violates the general assumption of all logics discussed herein that encryptions are not breakable by direct attack (to reveal either the plaintext or the key).

GNY Freshness Rules
All of these rules are derivable in SVO except F5 and

F6. F5 says that a principal's belief in the freshness of a private key follows from his belief in the freshness of its public cognate. F6 expresses the converse inference. There is no reason in practice to question these rules; however, there is also no harm in practice in leaving them out since public keys are usually long term and not distributed on line. They thus do not generally play a role in freshness considerations. F11 is only derivable in SVO assuming R6, which will be discussed shortly.

GNY Recognizability Rules
All of these rules are derivable in SVO except R6. This rule says that $P \models \phi(X)$ follows from $P \ni H(X)$. But, from the mere possession of $H(X)$, P should not form any beliefs about X; without X, he may not know that he is seeing $H(X)$ rather than some other message or even just a random bitstring. R6 as given in GNY is thus too strong, although perhaps only with respect to this intuition. If we replace the statement that P believes X is recognizable with a claim that X is recognizable by P we get a more reasonable conclusion. However, we have no formal means to directly represent this in either SVO or GNY. SVO does have the expressive capability to indicate that a principal recognizes a given bitstring as the same one that yielded the hash he received in a previous message, which appears to be the intended effect of R6. Recall that GNY only expresses recognizability in the context of belief, e.g., $P \models \phi(X)$, and this is the GNY formula for which we have provided an SVO translation. Indeed, as the above discussion shows, our treatment allows us to capture the effects of GNY recognizability with weaker logical rules.

GNY Message Interpretation Rules
We do not attempt to handle all of these, on general grounds of logical unwieldiness and inelegance. We make an admittedly arbitrary division by addressing only those rules containing less than five premises. Once appropriate translations have been made, these are derivable in SVO except for the second conclusion of I4: $P \models Q \mid\sim \{X\}_{-K}$. We saw no practical value of such a conclusion. Should this be incorrect, Q *said* $[X]_{K^{-1}}$ can be added to the consequent of axiom Ax4. Similar addition can be made to axiom Ax3. This logic remains sound with respect to the semantics given in §3. As mentioned earlier, some BAN logics assume message recovery from signatures. GNY does not actually even explicitly discuss signatures. I4 and I5 are meant to be used with public key encryption schemes such as RSA, where $\{\{X\}_{K^{-1}}\}_K = X$. In claiming that we can capture the reasoning of these rules, we are assuming in our translation that a more common scheme (for which message recovery is not possible) is being used rather than one such as they describe.

GNY Jurisdiction Rules
Like AT, SVO separates belief from everything else, including trust. This is useful (and perhaps the only way one is likely to maintain a model-theoretic semantics). The only jurisdiction rule (actually axiom) in SVO is the same as in AT, viz: P *controls* $\varphi \wedge P$ *says* $\varphi \supset \varphi$.

GNY J1 is taken directly from BAN's jurisdiction rule. BAN also has only one rule in this category. Nonetheless, BAN's rule is not derivable from the above nor valid in the semantics. This is no great loss since the only iterated beliefs we generally care about are derived from things that one principal says to another. In other words, the above axiom captures what we need from J1. BAN and GNY must express jurisdiction in terms of belief since that is their only way to capture a principal's actions in the current epoch. A more detailed discussion of this is given in [AT91], §3.2.

As Gong et al. say (p. 240) that J3 is just a special case of J2, we focus on J2.
(From $P \models Q \models\!\!\Rightarrow Q \models *$, $P \models Q \mid\sim (X \leadsto C)$, and $P \models \#X$, infer $P \models Q \models C$.) This rule introduces new notation not discussed elsewhere. '$P \models Q \models\!\!\Rightarrow Q \models *$' captures the idea that P believes Q to be honest (Q only says what he believes) and competent (Q understands the implications of what he says). This can be translated directly to the following SVO syntax expression: P *believes* $(((Q \textit{ says } X) \wedge (X \supset C)) \supset (Q \textit{ believes } C))$. The second premise of the rule can also be translated directly to SVO: P *believes* $((Q \textit{ said } X) \wedge (X \supset C))$. And, the third premise is the same in GNY and SVO, except for an irrelevant notational difference. Similarly, the conclusion of the rule is the same in GNY and SVO. So, the rule is entirely expressible within the SVO syntax. Furthermore, it is not only sound but an easy logical derivation in SVO.

B Relation to VO extensions

The first paper to introduce the capability to reason about key agreement, e.g., Diffie-Hellman exchanges, to a BAN-like logic is [vO93b]. Some of the notation and rules intoduced therein arise naturally in such protocols, but they are also applicable to shared and private key protocols as discussed in the above papers.

B.1 VO Notational Additions and Logical Rules

$A \overset{K-}{\longleftrightarrow} B$: K is A's *unconfirmed secret* suitable for B. No one aside from A and B and those they trust knows or could deduce K. This construct emphasizes, however, that while A knows K, B may or may not. This notation arises quite naturally when looking at key agreement protocols, such as Diffie-Hellman type key distributions, and is actually easy to capture in our semantics. Since '$A \overset{K}{\leftrightarrow} B$' simply means that K is a good key for A and B regardless of whether either of them knows this, we can actually define $A \overset{K-}{\longleftrightarrow} B$ in the SVO syntax: $(A \overset{K}{\leftrightarrow} B) \wedge (A \textit{ sees } K)$.

$A \overset{K+}{\longleftrightarrow} B$: K is A's *confirmed secret* suitable for B. A

knows K, and has received evidence confirming that B knows K. No parties other than A and B and those they trust know or can feasibly deduce K. This is a little trickier to capture in our semantics. For we must decide what it means for A to receive confirmation that B knows K. Let us consider a typical example of such confirmation in a protocol. Suppose B has just received the session key K and wants to confirm this to A. If she has sent him a nonce N_a earlier in the protocol run, a typical way for B to send confirmation is by encrypting N_a (or perhaps $N_a - 1$) with K and his own name and sending this to A. VO reasons about the key confirmation B sends to A in this example by introducing confirmation axioms, which we will discuss below when we come to the *confirm*(K) notation.

How would this key confirmation be handled using existing constructs in SVO? Consider an SVO analysis of a key distribution protocol where the above confirmation occurs. The standard practice in [BAN89] would be to idealize this in the protocol analysis by B sending to A $\{N_a, (A \overset{K}{\leftrightarrow} B), B\}_K$. In other words, the protocol idealization of B's sending such a message incorporates B saying that K is a good key for A and himself. But, notation of the form $A \overset{K}{\leftrightarrow} B$ is BAN's only way to express statements about a key. Using SVO notation we can make the more accurate interpretation of this message as $\{N_a, (B \textit{ sees } K), B\}_K$. Thus our premise set would include $A \textit{ believes } (A \textit{ received } \{N_a - 1, B\}_K \supset A \textit{ received } \{(N_a - 1, (B \textit{ sees } K), B\}_K)$. Given that A has the necessary beliefs about the freshness of N_a and the (unconfirmed) goodness of K we can derive the conclusion of the VO key confirmation rule (R32) within SVO. Thus, if we translate the syntax $A \overset{K+}{\longleftrightarrow} B$ as $A \textit{ believes } ((A \overset{K-}{\longleftrightarrow} B) \wedge (U \textit{ says } U \textit{ sees } K))$, where $U \neq A$, reasoning about key confirmation can be captured entirely within SVO. (Translating this fully back to the SVO syntax we get $A \textit{ believes } ((A \overset{K}{\leftrightarrow} B \wedge A \textit{ sees } K) \wedge (U \textit{ says } (U \textit{ sees } K)))$, where $U \neq A$.)[10]

The technique of the last paragraph allows us to capture key confirmation entirely without adding explicit confirmation syntax to SVO. However, there is a hidden informal assumption in such an approach. We can only use it if we systematically employ meta-rules for premise formation. Instead of explicitly using the confirmation axioms (C1–C3) of [vO93b] we must, in effect, always employ those axioms in premises of this type (i.e., receiver's interpretation premises). On the other hand, if we add the VO notation and rules, there is no need to give, e.g., A's interpretation of receiving $\{N_a\}_K$. We thus have a choice. On the one hand is a more streamlined logic and semantics accompanied by more assumptions about message interpretation, while on the other is a more complex logic and semantics accompanied by fewer such assumptions. By far the greatest source of confusion and misapplication of BAN to date has come from slipping dubious assumptions in (or leaving necessary assumptions out) during protocol idealization. The more formally explicit approach is thus safer, but either can be rigorously followed to the same practical effect. In the next paragraph we will discuss a proposal that combines the advantages of explicit axioms and a simpler logic.

confirm(K): *Current knowledge of K has been demonstrated.* We have been discussing the relative merits of capturing key confirmation via axioms and via direct translation to the syntax of SVO. If we choose to follow the latter route, then '*confirm*(K)' becomes irrelevant notation. The confirmation axioms make use of recognizability in the sense of GNY. Thus, if we wish to follow the former route, we will have to relativize '*confirm*(K)' in just the way that we relativized '$\phi(X)$' in appendix A.1. For convenience in the following discussion we introduce the syntactic shorthand $\phi_P(X) \equiv P \textit{ believes } P \textit{ sees } X$. (This would be intuitively too strong if $\phi_P(X)$ were understood as X is recognizable to P. The intuitive reading in what follows might better be rendered as P recognizes X, for which $P \textit{ believes } P \textit{ sees } X$ is a more acceptable approximation. In any case, the following discussion will ultimately obviate this notation.) The relativization is thus trivial notationally. For example, VO axiom C3 becomes

$$\textit{fresh}(K) \wedge \phi_P(H(K)) \supset \textit{confirm}_P(K)$$

We could use this to try to treat $\textit{confirm}_P(K)$ as a defined term following the axioms. But this raises some problems. Suppose we introduce the following definition (which encompasses C1, C2, and C3):

$$\textit{confirm}_P(K) \quad \equiv \quad \begin{array}{l} (\textit{fresh}(X) \wedge \phi_P(\{X\}_K)) \vee \\ (\textit{fresh}(X) \wedge \phi_P(\mathrm{MAC}_K(X)) \vee \\ (\textit{fresh}(K) \wedge \phi_P(H(K))) \end{array}$$

If we were then to try to apply this in VO rule R32, we would need to verify that $A \textit{ received } {*}\textit{confirm}_A(K)$. (Recall that VO follows GNY in using '$*$' to indicate that a message orginated elsewhere, rather than to indicate a message that may not be understood—as in SVO.) Unpacking the syntactic definition this would mean that $A \textit{ received } {*}((\textit{fresh}(X) \wedge \phi_P(\{X\}_K)) \vee (\textit{fresh}(X) \wedge \phi_P(\mathrm{MAC}_K(X)) \vee (\textit{fresh}(K) \wedge \phi_P(H(K))))$. But, since receiving does not distribute across disjunctions, this would never actually be satisfied. Actually this problem exists for R32 even before we attempt to give a definition: it is clear that in the condition $A \textit{ received } {*}\textit{confirm}_A(K)$, A is not meant to see a statement regarding freshness. Rather she is supposed to see

[10] For reasons that will soon become apparent, we will give a revised definition of '$A \overset{K+}{\longleftrightarrow} B$' below.

a statement that contains a fresh component. In addition there is the open endedness of the axiom list. These axioms were meant to capture three common ways of establishing key confirmation in practice, but others are possible. A fourth would simply involve sending the key K itself in a message; the message would have to be fresh somehow itself if the key K was not known to be fresh. (Note that in Diffie-Hellman key agreement, it is.) So, another axiom would be

$$\text{C4. } \phi_P(K) \wedge \mathit{fresh}(K) \supset \mathit{confirm}_P(K)$$

These and similar possibilities can all be represented in SVO by a single syntactic definition:

$$\mathit{confirm}_P(K) \equiv ((P \; \mathit{received} \; F(X,K) \wedge \phi_P(F(X,K)) \wedge (\mathit{fresh}(X) \vee \mathit{fresh}(K)))$$

Here F is a feasibly computable function, that is effectively one-one. This means it is infeasible to find any two pairs (X,K) mapping to the same value. F is required to be one-way (in the sense that encryptions, MACs, and cryptographic hash functions would be) if and only if it is important that K not be revealed by the confirmation process itself.[11] This also allows a more general definition of (data) confirmation (rather than key confirmation). Restricting confirmation to keys seems unnecessary, and it should not be a general constraint that data are not revealed through the confirmation process. Ways of confirming knowledge of information without revealing the information itself is the subject of a large area of research, namely zero-knowledge; this subject is beyond the scope of the present work. Note X can be null, and F could be the identity function, as in C4, the above axiom. We have incorporated 'P *received* $F(X,K)$' into the definition because confirmation is only relevant if someone receives it. Bringing this into the axiom itself avoids the problem of distributing *received* raised above. We can provide a similar definition to indicate that P has received confirmation from someone other than herself:

$$*\mathit{confirm}_P(K) \equiv (P \; \mathit{believes} \; P \; \mathit{received} \; F(X,K)) \wedge \neg(P \; \mathit{said} \; F(X,K)) \wedge (\mathit{fresh}(X) \vee \mathit{fresh}(K))$$

The definition just introduced has a number of advantages. It makes confirmation criteria explicit but constitutes no addition to SVO since it is eliminable, i.e., it can always be replaced by the longer expression that is purely in the language of SVO. (We have already dropped in this definition the notational shorthand of $\phi_P(F(X,K))$.) As just indicated, its application goes beyond the current context. It still requires that informal work be done, but the interpretation of protocol messages is as direct as it would be were we to use the axioms from [vO93b]. (As in our example of returning an encrypted nonce above, A's receipt of $\{N_a - 1, B\}_K$ need not be interpreted as receipt of $\{N_a - 1, (B \; \mathit{sees} \; K), B\}_K$.) The informal step is in determining whether or not this constitutes a function and functional arguments as stipulated in the axiom. But, this question is not subject to the same difficulties as when determining the intended meaning of a message. Here we need only make a determination based on mathematically rigorous criteria—up to the limits of the usual cryptographic assumptions made in protocol analysis.

Given the considerations of the last several paragraphs, we revise our definition of '$A \overset{K+}{\longleftrightarrow} B$'.

$$A \overset{K+}{\longleftrightarrow} B \equiv ((A \; \mathit{believes} \; A \overset{K-}{\longleftrightarrow} B) \wedge *\mathit{confirm}_A(K))$$

We now turn to notation for reasoning about public and private keys. The BAN notation to represent that K is A's public key is '$\overset{K}{\mapsto} A$'. It is simply assumed in BAN that the corresponding private key is kept secret. Notation for the private key, 'K^{-1}', is only used to indicate encryption using the key, e.g., $\{X\}_{K^{-1}}$. A's posession of K^{-1} is meant to be implicitly inferred from A *believes* $\overset{K}{\mapsto} A$. GNY introduces syntax for explicitly representing and reasoning about possession of private keys. Nonetheless, goodness of a private key is still meant to be inferred from a statement about the public key as in BAN, i.e., from $\overset{K}{\mapsto} A$. In [GS91], Gaarder and Snekkenes separate statements representing that A has associated a good public key K, viz: $\mathrm{PK}(A,K)$, from those representing that A has associated some good private key, viz: $\Pi(A)$. Thus the judgement about the quality of the private key is now associated with a statement about the private key, rather than being implied by a statement about the public key. In effect, this separates statements about the binding of a public key to a principal from statements about the quality of a principal's private key. Gaarder and Snekkenes separated these to reason about certificates binding a principal to a public key in the X.509 protocol separately from evaluating trust that the corresponding private key is kept secret. VO follows the developments of Gaarder and Snekkenes and also introduces distinct notation for public keys for signing, enciphering, and key agreement.

$\mathrm{PK}_\sigma(A,K)$: K is the *public signature verification key* associated with principal A.
$\mathrm{PK}_\sigma^{-1}(A)$: A's *private signature key* K^{-1} *is good.* Here K^{-1} corresponds to the public key K in $\mathrm{PK}_\sigma(A,K)$.[12]

[11] In confirming knowledge of K, the intention is that the key K itself is not revealed. However, in terms of formal definition, this is irrelevant—what is of import is confirmation only. If a key K is somehow compromised, whether in relation to key confirmation or otherwise, this may violate an assumption about key quality, but that should be treated distinctly from key confirmation.

[12] We are following convention here by using 'K^{-1}' to refer to a private signature key. Some schemes such as RSA can be used

Analogous definitions are made for enciphering ($\mathrm{PK}_\psi(A,K)$, $\mathrm{PK}_\psi^{-1}(A)$) and key agreement ($\mathrm{PK}_\delta(A,K)$, $\mathrm{PK}_\delta^{-1}(A)$). Unfortunately in the semantics of §3 we were unable to give truth conditions for all of these individually. We have reverted to grouping the binding of a public key together with the quality (secrecy) of the private key. We thus use '$\mathrm{PK}(A,K)$' to mean both that K is the public key associated with principal A *and* that the corresponding private key, K^{-1}, is good. If this is a loss, it is logically speaking a minor one. There are good reasons for separating the two notions. For, there are two distinct kinds of protocol failures here. On the one hand, the secrecy of a private key might be compromised. On the other hand, a principal A might be tricked into thinking that the wrong public key is bound to principal B. The distinction introduced by Gaarder and Snekkenes allows us to differentiate these failures. Nonetheless, the only logical use of the corresponding expressions occurs in their rule R13, where both proper binding and good private keys are premises of the rule. (Actually, what is required is belief therein, but this is aside.) This is similarly true for VO's rules. Thus, since both good public binding and good private keys are required for any logical use of these notions, it is sufficient to have notation that captures them together. (Nevertheless, we acknowledge that it would be nice to have the requirements syntactically separated for a more direct reflection of the nature of potential failures.)

Aside from the key confirmation axioms already discussed, VO introduces three new logical rules. (These are presented in appendix E.) They are all derivable in SVO, with the translations discussed above.

C GNY Rules

We present these GNY rules without any explanation of the rules or notation therein. Readers are referred to [GNY90] for details.

C.1 Rationality Rule

If $\dfrac{C1}{C2}$ is a rule, then for any principal P, so is $\dfrac{P \models C1}{P \models C2}$.

C.2 Being-Told Rules

T1 $\dfrac{P \triangleleft *X}{P \triangleleft X}$

T2 $\dfrac{P \triangleleft (X,Y)}{P \triangleleft X}$

T3 $\dfrac{P \triangleleft \{X\}_K,\ P \ni K}{P \triangleleft X}$

for both enciphering and signatures because of invertibility. This makes the notational choice quite natural. However, some signature schemes are not invertible, and for those schemes the notation is slightly deceptive.

T4 $\dfrac{P \triangleleft \{X\}_{+K},\ P \ni -K}{P \triangleleft X}$

T5 $\dfrac{P \triangleleft F(X,Y),\ P \ni X}{P \triangleleft Y}$

T6 $\dfrac{P \triangleleft \{X\}_{-K},\ P \ni +K}{P \triangleleft X}$

C.3 Possession Rules

P1 $\dfrac{P \triangleleft X}{P \ni X}$

P2 $\dfrac{P \ni X, P \ni Y}{P \ni (X,Y),\ P \ni F(X,Y)}$

P3 $\dfrac{P \ni (X,Y)}{P \ni X}$

P4 $\dfrac{P \ni X}{P \ni H(X)}$

P5 $\dfrac{P \ni F(X,Y),\ P \ni X}{P \ni Y}$

P6 $\dfrac{P \ni K,\ P \ni X}{P \ni \{X\}_K,\ P \ni \{X\}_K^{-1}}$

P7 $\dfrac{P \ni +K,\ P \ni X}{P \ni \{X\}_{+K}}$

P8 $\dfrac{P \ni -K,\ P \ni X}{P \ni \{X\}_{-K}}$

C.4 Freshness Rules

F1 $\dfrac{P \models \#(X)}{P \models \#(X,Y),\ P \models \#F(X)}$

F2 $\dfrac{P \models \#(X),\ P \ni K}{P \models \#(\{X\}_K),\ P \models \#(\{X\}_K^{-1})}$

F3 $\dfrac{P \models \#(X),\ P \ni +K}{P \models \#(\{X\}_{+K})}$

F4 $\dfrac{P \models \#(X),\ P \ni -K}{P \models \#(\{X\}_{-K})}$

F5 $\dfrac{P \models \#(+K)}{P \models \#(-K)}$

F6 $\dfrac{P \models \#(-K)}{P \models \#(+K)}$

F7 $\dfrac{P \models \phi(X),\ P \models \#(K),\ P \ni K}{P \models \#(\{X\}_K),\ P \models \#(\{X\}_K^{-1})}$

F8 $\dfrac{P \models \phi(X),\ P \models \#(+K),\ P \ni +K}{P \models \#(\{X\}_{+K})}$

F9 $\dfrac{P \models \phi(X),\ P \models \#(-K),\ P \ni -K}{P \models \#(\{X\}_{-K})}$

F10 $\dfrac{P \models \#(X),\ P \ni X}{P \models \#(H(X))}$

F11 $\dfrac{P \models \#(H(X)),\ P \ni H(X)}{P \models \#(X)}$

C.5 Recognizability Rules

R1 $\dfrac{P \models \phi(X)}{P \models \phi(X,Y),\ P \models \phi(F(X))}$

R2 $\dfrac{P \models \phi(X),\ P \ni K}{P \models \phi(\{X\}_K),\ P \models \phi(\{X\}_K^{-1})}$

R3 $\dfrac{P \models \phi(X),\ P \ni +K}{P \models \phi(\{X\}_{+K})}$

R4 $\dfrac{P \models \phi(X),\ P \ni -K}{P \models \phi(\{X\}_{-K})}$

R5 $\dfrac{P \models \phi(X),\ P \ni X}{P \models \phi(H(X))}$

R6 $\dfrac{P \ni H(X)}{P \models \phi(X)}$

C.6 Message Interpretation Rules

We present only I4, I6, and I7.

I4 $\dfrac{P \triangleleft \{X\}_{-K},\ P \ni +K,\ P \models \stackrel{+K}{\mapsto} Q,\ P \models \phi(X)}{P \models Q \mid\!\sim X,\ P \models Q \mid\!\sim \{X\}_{-K}}$

I6 $\dfrac{P \models Q \mid\!\sim X,\ P \models \#(X)}{P \models Q \ni X}$

I7 $\dfrac{P \models Q \mid\!\sim (X,Y)}{P \models Q \mid\!\sim X}$

C.7 Jurisdiction Rules

J1 $\dfrac{P \models Q \Rightarrow C,\ P \models Q \models C}{P \models C}$

J2
$$\frac{P \models Q \Rightarrow Q \models *,\ P \models Q \mid\!\sim (X \leadsto C),\ P \models \#(X)}{P \models Q \models C}$$

J3 $\dfrac{P \models Q \Rightarrow Q \models *,\ P \models Q \models Q \models C}{P \models Q \models C}$

D AT Rules and Axioms

We present these AT rules and axioms without explanation. Readers are referred to [AT91] for details.

There are two 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.

Axioms are all instances of tautologies of classical propositional calculus, and all instances of the following axiom schemata:

Believing

For any principal P and formulae φ and ψ,

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

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

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

Message Meaning

If $P \neq S$, then

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

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

Seeing

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

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

Saying

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

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

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

If ' *says* ' is substituted for ' *said* ' throughout in A12, A13, or A14, the result is also an axiom.

Jurisdiction

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

Freshness

A16. $fresh(X_i) \supset fresh(X_1, \ldots, X_n)$

A17. $fresh(X) \supset fresh(\{X\}_K)$

A18. $fresh(X) \supset fresh(\langle X \rangle_S)$

A19. $fresh(X) \supset fresh(`X')$

Nonce-Verification

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

Shared Keys and Secrets

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

A22. $R \stackrel{K}{\rightleftharpoons} R' \equiv R' \stackrel{K}{\rightleftharpoons} R$

E VO Rules

We present the three rules introduced in [vO93b] (in the original notation).

R30 $$\frac{A\ \mathit{has}\ \mathrm{PK}_\delta^{-1}(A),\ A\ \mathit{has}\ \mathrm{PK}_\delta(U)}{A\ \mathit{has}\ K}$$

where $K = f(\mathrm{PK}_\delta^{-1}(A), \mathrm{PK}_\delta(U))$.

R31 $$\frac{A \models \mathrm{PK}_\delta^{-1}(A),\ A \models \mathrm{PK}_\delta(B),\ A \models \mathrm{PK}_\delta^{-1}(B)}{A \models A \xleftrightarrow{K-} B}$$

where $K = f(\mathrm{PK}_\delta^{-1}(A), \mathrm{PK}_\delta(B))$.

R32 $$\frac{A \models A \xleftrightarrow{K-} B,\ A\ \mathit{sees}\ *\mathit{confirm}(K)}{A \models A \xleftrightarrow{K+} B}$$