On Unifying Some Cryptographic Protocol Logics

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

Paul C. van Oorschot*
Bell-Northern Research
P.O. Box 3511, Station C
Ottawa, Canada K1Y 4H7
(paulv@bnr.ca)

Abstract

We present a logic for analyzing cryptographic protocols. This logic encompasses a unification of four of its predecessors in the BAN family of logics, namely those given in [GNY90], [AT91], [vO93], and BAN itself [BAN89]. We also present a model-theoretic semantics with respect to which the logic is sound. The logic herein captures all of the desirable features of its predecessors and more; nonetheless, it accomplishes this with no more axioms or rules than the simplest of its predecessors.

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 [vO93]. (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 virtually all of 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. 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, 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 lan-

---

*also with: School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Canada K1S 5B6

[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.

guage 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 present a model of computation and a semantics for the language presented in §1. The remainder of the paper looks at the language and logic of GNY and VO in comparison to SVO. In particular we consider how to capture in SVO the linguistic expressibility and logical derivability of GNY and VO. 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 space limitations preclude an adequate presentation of the full Abadi-Tuttle semantics. We therefore make comparative comments at appropriate points throughout §§1 and 2.

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 idealized 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 $\mathcal{T}_0$ of primitive terms containing a number of disjoint sets of constant symbols representing principals, shared keys, public keys, private keys, numerical constants, etc. Building recursively on $\mathcal{T}_0$, we have n-ary function symbols representing functions of n variables, for finite n, e.g., simple arithmetical functions, encryption, etc. In addition to these is a set of primitive proposition constants. These represent atomic propositions, which take the value true or false. The full set of terms is called '$\mathcal{T}$'. We actually require two formal languages, one for messages and one for formulae. Only formulae can be true or false or have principal's beliefs attributed to them. On the other hand, some messages are not formulae, e.g., a message consisting of a name and a nonce. 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, $(X_1, \ldots, X_n)$, and encryptions, $\{X\}_K$),
- φ is a message if φ is a formula.

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

- p is a formula if p is a primitive proposition,
- $\neg\varphi$ and $\varphi \wedge \psi$ are formulae if φ and ψ are formulae (other connectives are definable in the usual manner),
- P *believes* φ and P *controls* φ are formulae when φ is a formula and P is a principal,
- 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,
- $P \overset{K}{\leftrightarrow} Q$, $\mathrm{PK}(P, K)$, and P *has* K are formulae when P and Q are principals and K is a key.

Most of the expressions just given either are standard notation[2] or should be intuitively clear. We give a brief intutive 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 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. (We actually have distinct notation for public keys for encryption, signature, and key agreement, viz: $\mathrm{PK}_\psi(P, K)$, $\mathrm{PK}_\sigma(P, K)$, and $\mathrm{PK}_\delta(P, K)$, respectively. These will be discussed below in §2.2.)

A few more notes on notation: Typically '$\{X\}_K$' is meant to refer to transformations of X using K. We mean specifically to include digital signatures under this notation as well as shared and public key encryption. We may occasionally write '$\{X^P\}_K$' to indicate that a message, X, is from P. We do not simply write it as an encrypted from field since other mechanisms may be used, e.g., direction bits. P is, however, written inside the scope of the encryption to indicate that it is considered bound to the message in a secure manner. We assume principals to be competent (though not necessarily honest) in setting from fields.

---

[2] We 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., '$A \longrightarrow B$'.

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 function. It does not assume that either the function or its complement (inverse) is computable in practice. 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 Logic

Our logic 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.[3] '$\Gamma \vdash \varphi$' means that φ is derivable from the set of formulae Γ (and the axioms). '$\vdash \varphi$' means that φ is a theorem, i.e., derivable from axioms alone. We describe derivability (i.e. proofs) below in §1.3. 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,

1. $P \textit{ believes } \varphi \wedge P \textit{ believes } (\varphi \supset \psi) \supset P \textit{ believes } \psi$

2. $P \textit{ believes } \varphi \supset P \textit{ believes } (P \textit{ believes } \varphi)$

Axiom 1 says that a principal believes all that logically follows from his beliefs. Axioms 2 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.

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

4. $(\mathrm{PK}_\sigma(Q, K) \wedge R \textit{ received } \{X\}_{K^{-1}}) \supset Q \textit{ said } X$

Recall that '$\mathrm{PK}_\sigma(Q, K)$' says that K is the public signature verification key for Q. Precise meaning is set out in §2.2. 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. A key can be applied (by anyone who has it) to any bitstring to yield another bitstring. Since the language does not represent arbitrary bitstrings, we avoid attributing to principals the inappropriate decryption of any such bitstring.

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

5. $((\mathrm{PK}_\delta(P, K_p) \wedge (\mathrm{PK}_\delta(Q, K_q)) \supset P \xleftrightarrow{K_{pq}} Q$

Here $K_{pq} = f(K_p, K_q^{-1}) = f(K_q, K_p^{-1})$ where f is some key-agreement function as in Diffie-Hellman key exchange. Recall that '$\mathrm{PK}_\delta(R, K)$' says that K is the public key-agreement key for R and implies that K_r^{-1} remains secret. Precise meaning is set out in §2.2.

Receiving A principal receives the concatenates of received messages and decryptions with available keys.

6. $P \textit{ received } (X_1, \ldots, X_n) \supset P \textit{ received } X_i$

7. $(P \textit{ received } \{X\}_K \wedge P \textit{ has } \widetilde{K}) \supset P \textit{ received } X$

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 §2.2.

8. $P \textit{ received } X \supset P \textit{ sees } X$

9. $P \textit{ sees } (X_1, \ldots, X_n) \supset P \textit{ sees } X_i$

10. $(P \textit{ sees } X_1 \wedge \ldots \wedge P \textit{ sees } X_n) \supset$
    $(P \textit{ sees } F(X_1, \ldots, X_n))$

Here F is any function computable in practice by P. There is no axiom for seeing corresponding to axiom 7 for receiving, i.e., $(P \textit{ sees } \{X\}_K \wedge P \textit{ has } \widetilde{K}) \supset P \textit{ sees } X$. Such an axiom is a special case of axiom 10, where F is the application of $\widetilde{K}$ to $\{X\}_K$, and axiom 20 $(P \textit{ has } \widetilde{K} \equiv P \textit{ sees } \widetilde{K})$.

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.

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

12. $(P \textit{ received } F(X) \wedge P \textit{ believes } P \textit{ sees } X) \supset$
    $P \textit{ believes } P \textit{ received } F(X)$

Here F is any effectively one-one function, and 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. The meaning of these axioms

---

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

is made clear by the truth conditions for belief set out in §2.2. These axioms capture what we want of GNY's recognizability. They also serve as a replacement for A11 of AT. Abadi and Tuttle have noted that the axiom was unsound as presented, but that they have a revision that is sound.[4] Note that the converse of axiom 11 is a theorem, following from axiom 1 and axiom 10 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.

13. $P \textit{ said } (X_1, \ldots, X_n) \supset (P \textit{ said } X_i \wedge P \textit{ sees } X_i)$

14. $P \textit{ says } (X_1, \ldots, X_n) \supset$
 $(P \textit{ said } (X_1, \ldots, X_n) \wedge P \textit{ says } X_i)$

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

15. $(P \textit{ controls } \varphi \wedge P \textit{ says } \varphi) \supset \varphi$

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.

16. $\textit{fresh}(X_i) \supset \textit{fresh}(X_1, \ldots, X_n)$

17. $\textit{fresh}(X_1, \ldots, X_n) \supset \textit{fresh}(F(X_1, \ldots, X_n))$

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

18. $(\textit{fresh}(X) \wedge P \textit{ said } X) \supset P \textit{ says } X$

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

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

Having A principal has a key iff he sees it.

20. $P \textit{ has } K \equiv P \textit{ sees } K$

1.3 Syntactic Analysis

In this section we give a brief description of the syntactic protocol analysis technique, which is similar to the techniques given in [BAN89] and [AT91]. The first step of this technique is *protocol idealization*. Consider a protocol step in which a key server S distributes a key to principal A for the purpose of talking with B. A typical example might thus be written,

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

This means that S has sent the following to A (all encrypted with K_{as}, a key shared by A and S): a timestamp, T_s, B's name, and the session key K_{ab}. In our language we have already abstracted away from bitstrings sent in messages to the elements of the language that those bitstrings represent. But, we must go still further. For, even if we represent that S has sent K_{ab} to A, we have not reflected that by this transmission S asserts K_{ab} to be a good key for a session between A and B.[5] This is done via protocol idealization.[6] The above protocol step is rewritten in the idealized form:

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

Once we have the idealized protocol, we write down corresponding formulae in the logic following a procedure called 'protocol annotation' in [BAN89]. We will use the formulae generated by annotation as the premise set in proofs of protocol goals. To generate this set we first write down the initial assumptions. These are things that are assumed to be true before the start of the protocol. For example, $A \overset{K_{as}}{\longleftrightarrow} S$ and A *believes* $(A \overset{K_{as}}{\longleftrightarrow} S)$ would probably be needed as initial assumptions in order to determine anything useful from the above protocol step. If a protocol analysis assumes that a principal P comprehends a message X, we require that this comprehension be explicitly set out in the initial assumptions by P *believes* P *sees* X. We may also add to the premise set Q *received* X for any step in the protocol, $P \longrightarrow Q : X$. Finally, we may add to the premise set P *sees* X for any protocol step in which P generates X. Typically, this X will be a nonce or a key or some such thing. (For example, the above protocol step justifies adding S *sees* K_{ab} to the premise set.)

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 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.[7] Of course, in AT and SVO necessitation must always

---

[4] Personal communication.

[5] It is merely a mnemonic device that the distributed key in this case is usually labelled 'K_{ab}'.

[6] Unlike in [BAN89], we do not assume that cleartext is left out of the idealized protocol. As first noted in [GKSG91], such omission can sometimes create problems.

[7] Our choice to characterize proofs in this way has important repercussions for other features of the logic. In [AT91] it was necessary for analysis to restrict consideration to "good" runs where, e.g., initially held beliefs are true, where negations do not occur within belief operators in initially held beliefs, etc. We need place no such restrictions. We defer discussion because of space.

be restricted to theorems: we should not generally infer that each principal believes all the assumptions contained in the premise set. A typical goal of, e.g., a key distribution protocol would be that one of the principals believe that the distributed key is good for communication with the other.

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. Problems arising from initial assumptions, as in the Nessett protocol [Nes90], are thus addressible using these logics. (Cf. [Syv92] for a detailed discussion.)

2 Semantics

2.1 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 $\mathit{send}(X, G)$, $\mathit{receive}()$, and $\mathit{generate}(X)$ respectively. One can send and receive any message, but one can only generate primitive terms, i.e., members of $\mathcal{T}_0$. 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., $\mathit{receive}(X)$. 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. §2.2 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. They include encryptions and decryptions with available keys as well as other functions the principal may perform, e.g., hashes, signatures, arithmetical functions, etc. 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 *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, and (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. 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 received, newly generated, or initially available to P_i implicitly defines a set of *seen messages* for him at that point. This consists of the messages just mentioned plus all the messages he can recursively produce from those messages via his available transformations or by creating formulae from seen messages. (E.g., P *has* K, P *says* X, etc.) The *said messages* are somewhat more restricted; we cannot hold a principal responsible for saying everything that is derivable by him from things he said. 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. 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 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 of said submessages of all messages that P has sent in r through time t. We further restrict our model 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 $r_i(t)$. Relatedly, the set of available transformations for a given principal in a single run is monotonically nondecreasing over time.

2.2 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, $\mathcal{R}$ and an interpretation π that maps each proposition constant $p \in \mathcal{T}$ to a set of points $\pi(p)$, intuitively, those points at which p is true. 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).

Primitive Propositions and Logical Connectives

$(r, t) \models p$ iff $(r, t) \in \pi(p)$,

$(r, t) \models \varphi \wedge \psi$ 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 §2.1.

Seeing and Having

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

iff X is in the set of seen messages for P at (r, t), as defined in §2.1. Truth conditions for PhasK are the same, except that K can only be a key.

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 essentially the same as in [AT91]:

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

iff, for all k', $(r, k') \models R \ \mathit{said}\ \{X\}_K$ implies either $(r, k') \models R \ \mathit{received}\ \{X\}_K$ or $R \in \{P, Q\}$.

'$\mathrm{PK}(P, K)$' means both that K is the 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 will also use '$\mathrm{PK}^{-1}(P)$' not to express any proposition, but simply to refer to P's private key in the absence of a specific name. We similarly use '$\mathrm{PK}(P)$' to refer to P's 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 is for signature keys.

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

iff, and all t', $(r, t') \models Q \ \mathit{received}\ \{X\}_{K^{-1}}$ 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 $Q = P$.

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

---

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

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

iff for all t', (1) for some Q, $K_{pq} = f(K^{-1}, \mathrm{PK}_\delta(Q))$ implies $(r,t') \models P \xleftrightarrow{K_{pq}} Q$; and, (2) for all R, $K_{pr} = f(K^{-1}, \mathrm{PK}_\delta(R))$ and $(r,t') \not\models P \xleftrightarrow{K_{pr}} R$ implies, for all U, $K_{ur} = f(\mathrm{PK}_\delta^{-1}(U), \mathrm{PK}_\delta(R))$ implies $(r,t') \not\models U \xleftrightarrow{K_{ur}} R$. (Here f is some agreement function such as that in Diffie-Hellman key agreement. As with other encryption algorithms/functions in protocol analysis, we assume f is strong. In other words, attacks based on properties of the function not specified here are deemed beyond the scope of our analysis.) The first clause guarantees that there is someone with whom P can form a good key. The second clause guarantees that anyone with whom P cannot form a good key cannot form a good key with anybody. 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 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 K_{pq}. 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.

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, though they may be discernible from the one he is in. (This is because he may be mistaken about which state he is in.) 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. 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. We determine the set of *comprehended messages* for a given principal P_i at a given point (r,t) as follows.

Of the seen messages for P_i at (r,t), include in the comprehended messages all primitive terms of $\mathcal{T}_0$ and all proposition constants, also any formulae of the form $P \overset{K}{\leftrightarrow} Q$, $\mathrm{PK}(P,K)$, or P *has* K. The result is the basis for the set of comprehended messages. We can then recursively add seen messages to the basis set. If X is in the comprehended set and P is in the comprehended set, then any of the following are added (from the seen messages): P *sees* X, P *received* X, P *says* X, P *said* X, or *fresh*(X). Similarly, if P and a formula φ are in the available set, then P *believes* φ and P *controls* φ are comprehended. Any seen compound formula is comprehended if its subformulae are comprehended. We introduce new notation for concatenated formulae where some, but not all, of the concatenates are comprehended. If $(X_1, \ldots, X_n)$ is a seen message, then the result of replacing any X_i with $*$ so that only comprehended messages and $*$s appear in the concatenated message is comprehended. Finally, if (1) F is any effectively one-one function, and either F or $\widetilde{F}$ is computable in practice by P, (2) X is comprehended, and (3) $F(X)$ is a seen message, then $F(X)$ is comprehended.

Define *comprehension*$(P,(r,t),(r',t'))$ to be the set of messages that results from the applying the available transformations for P in (r,t) to the seen messages for P in (r',t') according to the procedure just given for producing the set of comprehended messages for P (at a single world). Thus the comprehended messages for P at (r,t) are exactly the members of *comprehension*$(P,(r,t),(r,t))$. The possibility relation $\sim_i$ for a principal P_i in state (r,t) is defined by

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

iff, local histories in $r_i(t)$ and $r'_i(t')$ are the same and *comprehension*$(P,(r,t),(r,t)) =$ *comprehension*$(P,(r,t),(r',t')) \subseteq$ *comprehension*$(P,(r',t'),(r',t'))$.[9] The second clause implies that a principal should consider possible those worlds that look the same and where his discernment capabilities are at least as great as they actually are. This does not mean that he can tell what any further capabilities would be.

We can now give truth conditions for belief formulae:

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

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

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

---

[9] Those who may have been wondering why SVO has no negative introspection axiom (axiom A3 in AT) should note that this relation is not euclidean. (Nor do we wish it to be.)

2.3 Soundness

Theorem 2.1 If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$. (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.)

Proof: (Sketch) This is a typical tedious soundness proof: 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 §2.2. As space is limited, we prove only validity of axiom 4 as an example.

$(\mathrm{PK}_\sigma(Q, K) \wedge R \textit{ received } \{X\}_{K^{-1}}) \supset Q \textit{ said } X$

This is a conditional, hence true at a point (r, t) if $(r, t) \not\models (\mathrm{PK}_\sigma(Q, K) \wedge R \textit{ received } \{X\}_{K^{-1}})$ or $(r, t) \models Q \textit{ said } X$. If the antecedent is false at (r, t) then, the conditional is true. If the antecedent is true at (r, t), then both of its conjuncts are true there. But, $(r, t) \models (\mathrm{PK}_\sigma(Q, K)$ implies that if $(r, t) \models R \textit{ received } \{X\}_{K^{-1}}$, then $(r, t) \models Q \textit{ said } X$. So axiom 4 is true at all worlds (r, t).

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, $\Gamma \models P \textit{ believes } \psi$. □

3 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.

3.1 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 $P \mid\equiv \phi(X)$ in GNY as P *believes* P *sees* X.

$P \triangleleft *X$: P is told a formula that he did not convey previously in the current run. This is captured in SVO as '$(P \textit{ received } X) \wedge \neg(P \textit{ says } X)$'. 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.

3.2 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 $P \mid\equiv C2$ from $P \mid\equiv C1$. It falls out of the modus ponens rule and axiom 1.

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 $P \mid\!\equiv \phi(X)$ 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. 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 represent this in either SVO or GNY. SVO does not 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. But, this is not a very serious limitation. Generally we would like to have such recognition only when the bitstring has some meaning to the recipient, i.e., corresponds to an element of the language that he comprehends. This type of recognition is captured in SVO via axiom 12.

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: $P \mid\!\equiv Q \mid\!\sim \{X\}_{-K}$. 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 4. (Similar addition can be made to axiom 3.) This logic remains sound with respect to the semantics given in §2.

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 $P \mid\!\equiv Q \mid\!\Rightarrow Q \mid\!\equiv *$, $P \mid\!\equiv Q \mid\!\sim (X \leadsto C)$, and $P \mid\!\equiv \#X$, infer $P \mid\!\equiv Q \mid\!\equiv C$.) This rule introduces new notation not discussed elsewhere. '$P \mid\!\equiv Q \mid\!\Rightarrow Q \mid\!\equiv *$' 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* $(((Q$ *says* $X) \wedge (X \supset C)) \supset (Q$ *believes* $C))$. The second premise of the rule can also be translated directly to SVO: P *believes* $((Q$ *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.

4 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 [vO93]. 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.

4.1 VO Notational Additions and Logical Rules

$A \xleftrightarrow{K-} B$: 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 $A \xleftrightarrow{K-} B$ in the SVO syntax: $(A \overset{K}{\leftrightarrow} B) \wedge (A\ \mathit{has}\ K)$.

$A \xleftrightarrow{K+} B$: K is A's *confirmed secret* sutiable 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 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 $\mathit{confirm}(K)$ 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 idealization of this message as $\{N_a, (B\ \mathit{has}\ K), B\}_K$. Following this idealization procedure, we could idealize A's receipt of the message we have been discussing as '$A\ \mathit{received}\ \{(N_a, (B\ \mathit{has}\ K), B\}_K$'. 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 $A \xleftrightarrow{K+} B$ as $A\ \mathit{believes}\ ((A \xleftrightarrow{K-} B) \wedge (U\ \mathit{says}\ U\ \mathit{has}\ K))$, where $U \neq A$.[10] Reasoning about key confirmation can be captured entirely within SVO. (Translating this fully back to the SVO syntax we get $A\ \mathit{believes}\ ((A \overset{K}{\leftrightarrow} B \wedge A\ \mathit{has}\ K) \wedge (U\ \mathit{says}\ (U\ \mathit{has}\ K)))$, where $U \neq A$.)

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 metarules for idealization. Instead of explicitly using VO confirmation axioms we must, in effect, always employ those axioms in protocol idealization. But, if we add the VO notation and rules, the idealization of the confirmation message is the same as its representation in the concrete protocol. (In other words $\{N_a\}_K$ is idealized simply as $\{N_a\}_K$.) We thus have a choice. On the one hand is a more streamlined logic and semantics accompanied by a more complex analysis procedure, while on the other is a more complex logic and semantics accompanied by a more straightforward analysis procedure. 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, clearer idealization, and a simpler logic.

$\mathit{confirm}(K)$: *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 '$\mathit{confirm}(K)$' becomes irrelevant notation. The axioms make use of recognizability in the sense of GNY. Thus, if we wish to follow the former route, we will have to relativize '$\mathit{confirm}(K)$' in just the way that we relativized '$\phi(X)$' in §3.1. For convenience in the following discussion we introduce the syntactic shorthand $\phi_P(X) \equiv P\ \mathit{believes}\ P\ \mathit{sees}\ X$. The relativization is thus trivial notationally. For example, VO axiom C3 becomes

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

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

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

If we were then to try to apply this in VO rule R32, we would need to verify that $A\ \mathit{received}\ {*}\mathit{confirm}_A(K)$. Unpacking the syntactic definition this would mean that $A\ \mathit{received}\ *((\mathit{fresh}(X) \wedge \phi_P(\{X\}_K)) \vee (\mathit{fresh}(X) \wedge \phi_P(\mathrm{MAC}_K(X)) \vee (\mathit{fresh}(K) \wedge \phi_P(H(K))))$. 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 $A\ \mathit{received}\ *\mathit{confirm}_A(K)$, A is not meant to see a statement regarding freshness. Rather she is supposed to see a statement that contains a fresh component. In addition there is the open endedness of the

---

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

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:

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

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:

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

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. As just indicated, its application goes beyond the current context. It still requires that informal work be done, but the idealization of the protocol is as direct as it would be were we to use the axioms from [vO93]. (As in our example of returning an encrypted nonce above, $\{N_a\}_K$ is still idealized as $\{N_a\}_K$.) 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 requiring confirmation judgements in protocol idealization. There we are required to determine the intended meaning of a message (fragment). 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 '$A \xleftrightarrow{K+} B$'.

$$A \xleftrightarrow{K+} B \equiv ((A \textit{ believes } A \xleftrightarrow{K-} B) \wedge *\mathit{confirm}_A(K))$$

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 introduce 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]

Analogous definitions are made for enciphering ($\mathrm{PK}_\psi(A,K)$, $\mathrm{PK}_\psi^{-1}(A)$) and key agreement

---

[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 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.

$(\mathrm{PK}_\delta(A, K),\ \mathrm{PK}_\delta^{-1}(A))$. Unfortunately in the semantics of §2 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 C.) They are all derivable in SVO, with the translations discussed above.

5 Conclusions and Further Study

> A formal method that tries to cover all the features of cryptographic protocol analysis is like a Swiss Army knife—not a terribly good instance of any of the tools it contains.
>
> —Roger Needham

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 as simple to use or simpler than any of those from which it is derived; yet it is more expressive than any of them.

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.

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]). In [Syv93a] a temporal version of BAN logic is presented that allows one to express general criteria for freedom from replays. It does not give a general means for detecting such replays. Thus it is only a first step; nonetheless, the introduced temporal operators are necessary if one is to even express such criteria in a BAN-like logic. That logic is sound with respect to the semantics presented in this paper. In fact, fully integrating it into the logic we have given is simply a matter of adding five axioms and a rule.

We also have yet to explore 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 logics. Nonetheless, we have produced a unified BAN-like logic that captures the features of four other logics. We have approached this from the perspective of having an integrated model. Thus, unlike a Swiss Army knife, our work is more than a collection of tools. Indeed, we believe it to be a better instance of all the tools it contains.

Acknowledgements

We would like to thank the anonymous referees for helpful comments and suggestions.

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.

[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.

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

[GKSG91] V.D. Gligor, R. Kailar, S. Stubblebine, and L. Gong. Logics for Cryptographic Protocols — Virtues and Limitations. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 219–226. IEEE Computer Society Press, Los Alamitos, California, 1991.

[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.

[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.

[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.

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

[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.

[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.

[vO93] 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.

A GNY Rules

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

A.1 Rationality Rule

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

A.2 Being-Told Rules

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

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

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

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

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

A.4 Freshness Rules

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

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

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

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

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

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

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

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

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

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

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

A.5 Recognizability Rules

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

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

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

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

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

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

A.6 Message Interpretation Rules

We present only I4, I6, and I7.

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

I6 $$\frac{P \mid\equiv Q \mid\sim X,\ P \mid\equiv \#(X)}{P \mid\equiv Q \ni X}$$

I7 $$\frac{P \mid\equiv Q \mid\sim (X,Y)}{P \mid\equiv Q \mid\sim X}$$

A.7 Jurisdiction Rules

J1 $$\frac{P \mid\equiv Q \mid\Rightarrow C,\ P \mid\equiv Q \mid\equiv C}{P \mid\equiv C}$$

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

J3 $$\frac{P \mid\equiv Q \mid\Rightarrow Q \mid\equiv *,\ P \mid\equiv Q \mid\equiv Q \mid\equiv C}{P \mid\equiv Q \mid\equiv C}$$

B 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 $\vdash P\ \textit{believes}\ \varphi$.

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. $P\ \textit{believes}\ \varphi \wedge P\ \textit{believes}\ (\varphi \supset \psi) \supset P\ \textit{believes}\ \psi$

A2. $P\ \textit{believes}\ \varphi \supset P\ \textit{believes}\ (P\ \textit{believes}\ \varphi)$

A3. $\neg(P\ \textit{believes}\ \varphi) \supset P\ \textit{believes}\ (\neg(P\ \textit{believes}\ \varphi))$

Message Meaning

If $P \neq S$, then

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

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

Seeing

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

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

A9. $P\ \textit{sees}\ \langle X^Q\rangle_S \supset P\ \textit{sees}\ X$

A10. $P\ \textit{sees}$ 'X' $\supset P\ \textit{sees}\ X$

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

Saying

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

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

A14. $P\ \textit{sees}$ 'X' $\wedge \neg P\ \textit{sees}\ X \supset P\ \textit{said}\ X$

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

Jurisdiction

A15. $P\ \textit{controls}\ \varphi \wedge P \textit{says} \varphi \supset \varphi$

Freshness

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

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

A18. $\textit{fresh}(X) \supset \textit{fresh}(\langle X\rangle_S)$

A19. $\textit{fresh}(X) \supset \textit{fresh}(\text{`}X')$

Nonce-Verification

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

Shared Keys and Secrets

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

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

C VO Rules

We present the three rules introduced in [vO93] (in the original notation).

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

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

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

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

R32 $$\frac{A \mid\equiv A \xleftrightarrow{K-} B,\ A\ \textit{sees}\ *\textit{confirm}(K)}{A \mid\equiv A \xleftrightarrow{K+} B}$$