The Logic of Authentication Protocols*

Paul Syverson[1] and Iliano Cervesato[2]

[1] Center for High Assurance Computer Systems, Naval Research Laboratory
Washington, DC 20375, USA
syverson@itd.nrl.navy.mil
[2] Advanced Engineering and Sciences Division, ITT Industries, Inc.
2560 Huntington Avenue, Alexandria, VA 22303, USA
iliano@itd.nrl.navy.mil

1 Introduction

The rationale of authentication has been a topic of study for about a decade and a half. First attempts at formal analysis of authentication protocols were not using logics per se, but were certainly logical. Millen's *Interrogator* [Mil84, MCF87] was a Prolog based tool specifically designed for authentication protocol analysis that functioned essentially as a special purpose model checker. Kemmerer used the general purpose formal specification language *Ina Jo* and an accompanying symbolic execution tool *Inatest* to specify and analyze protocols [Kem87].

We will focus on logics of authentication, beginning with BAN [BAN89a]. However, we will not only be discussing logics per se. We will also be looking at the 'rhyme and reason' of authentication, the attempts to formalize and define notions of authentication and to apply these. Thus, we will also be considering the logic of authentication in a broader sense.

We will not discuss (except incidentally) other formal methods that have been applied to authentication. In particular, we will not be describing process algebras, automata, automated tools such as theorem provers or model checkers. Some of these other approaches are discussed elsewhere in this volume. The remainder of this section will provide background on authentication protocols and introduce a running example.

1.1 Background on Authentication Protocols

In this section we present basic background on the concepts of authentication and its building blocks in cryptography. If every device communicating on behalf of a person or other entity shared a secret key with every other such device, and these keys were never compromised, canceled, unsubscribed, or otherwise

* This paper is based on a course Syverson taught at the 1st International School on Foundations of Security Analysis and Design (FOSAD'00) in Bertinoro, Italy in September 2000. Cervesato was a student there. The work of the first author was supported by ONR. The work of the second author was supported by NSF grant INT98-15731 "Logical Methods for Formal Verification of Software" and by NRL under contract N00173-00-C-2086.

R. Focardi and R. Gorrieri (Eds.): FOSAD 2000, LNCS 2171, pp. 63–137, 2001.

expired, then basic authentication protocols *might* be unnecessary. Clearly this is not even remotely the case. It has thus long been recognized that there must be some mechanism by which principals that do not share such a secret key, and may not even have any knowledge of each other beyond possibly an identifier, can establish a key for a secure communication session.

An *authentication protocol* is an exchange of messages having a specific form for authentication of principals using cryptographic algorithms. They typically have additional goals such as the distribution of session keys. *Symmetric-key cryptography* (also called *secret-key cryptography*) relies on the same key for both encryption and decryption. Classic examples include the data encryption standard, DES, and its recent successor, AES, the advanced encryption standard. (More details about cryptography can be found in any number of books [MvOV97, Sch96, Sti95].) *Public-key cryptography* is encryption and decryption using different keys (also called asymmetric cryptography). The most well known example is RSA. A *public key* is so-called because it is generally available to anyone. Corresponding to the public key is a *private key*, stereotypically known only to one principal. The private key is used to decrypt the message. Because it is uniquely bound to an individual a private key can also be used for a *digital signature* on a message. Typically, different keys and different algorithms are used for decryption and digital signatures. For digital signatures, the public key is used to verify that the signature is that of the principal bound to the public key. Binding is usually accomplished by means of a certificate, typically a message asserting such binding, containing an indicator of timeliness and signed by a well-known trusted principal (server).

Security protocols may have any number of intended purposes. Some exotic examples are voting, fair exchange of goods or contracts, non-repudiation, and anonymous communication. We will focus on authenticated establishment of session keys, which is typically necessary for the running of security protocols for most other purposes. Authentication is essentially assurance of who you are talking to. This can be made more specific in any number of ways: for example, you may want to make sure that those obtaining a session key are who they say they are, make sure that someone who has the key is currently on line, make sure that the principal you think has the key does have it, make sure that the principal with whom you think you share the key also thinks he is sharing it with you, etc. We will go into more detail on these points in Section 4. For now the basic intuition should suffice.

If a protocol is used for some security purpose, this implies an adversary against which the protocol is secure. The standard adversary for formal analysis of security protocols was introduced by Dolev and Yao in 1983 and is commonly known as the Dolev-Yao adversary [DY83]. It is a very strong adversary, much stronger than is typically assumed for secure distributed computation as in, e.g., Byzantine agreement. In the Dolev-Yao case, all messages sent from any honest principal to any other must pass through the adversary. The adversary can read, alter, and redirect any and all messages. However, encryption is treated as a black box. The adversary can only decrypt a message if she has the right keys.

She can only compose new messages from keys and messages that she already possesses. In particular, she cannot perform any statistical or other cryptanalytic attacks. Other common terms for the adversary include: attacker, penetrator, spy, intruder, enemy, and eavesdropper. Eavesdroppers are typically considered as only passive. But, any adversary is often referred to as 'Eve'.

1.2 Running Example

In this section, we introduce an example of an authentication protocol that will also be discussed in later sections. Eve's honest counterparts are traditionally named 'Alice' and 'Bob'. The other main principal in this protocol is the server. Alice and Bob are assumed to share keys (typically called 'long-term keys') with the server. Besides the obvious symbols for Alice (A), Bob (B), the server (S), and the keys they share (k_{AS}, k_{AS}, k_{AB}, etc.), the protocol introduces us to nonces, i.e., random unpredictable values generated by a principal and included in messages so that she can tell any messages later received and containing her nonce must have been produced after she generated and sent the nonce. A nonce generated by Alice is written 'n_A'. The session key that the server generates for Alice and Bob is k_{AB}. Encryption of a message, M, using key k is written '$\{M\}_k$'.

Protocol 1 (Needham-Schroeder Shared-Key) [NS78]

Message 1	$A \rightarrow S:$	A, B, n_A
Message 2	$S \rightarrow A:$	$\{n_A, B, k_{AB}, \{k_{AB}, A\}_{k_{BS}}\}_{k_{AS}}$
Message 3	$A \rightarrow B:$	$\{k_{AB}, A\}_{k_{BS}}$
Message 4	$B \rightarrow A:$	$\{n_B\}_{k_{AB}}$
Message 5	$A \rightarrow B:$	$\{n_B - 1\}_{k_{AB}}$

In this protocol, Alice indicates to the server that she would like to talk to Bob and includes a nonce, n_A. The server S sends her a message encrypted with the key they share. This message contains her nonce n_A (so that she knows the message is fresh), Bob's identifier (so she knows that this is indeed for a session between her and Bob), the session key k_{AB}, and an encrypted submessage $\{k_{AB}, A\}_{k_{BS}}$ to be forwarded to Bob. Alice decrypts this message and forwards the submessage to Bob. He decrypts it and sends Alice a nonce n_B encrypted with the session key to show that he has the session key and to check that she does. She decrypts the message, subtracts one from the nonce, re-encrypts, and sends it back to Bob, completing the protocol. In the next section, we will set out a logic for analyzing protocols such as this.

1.3 Organization

In Section 2 we will describe BAN logic, its rules and some limitations and revisions. In Section 3 we will describe one of BAN's successors, SVO. Both of those

sections will include an analysis of the running example. We will then proceed in Section 4 to the 'rhyme and reason', presenting formal and informal goals of authentication and attempts to define it. We will turn to some of the other aspects of authentication protocols in Section 5, where we will look at design principles and properties that arise from them, such as fail-stop properties. We will also look at ways in which these have been combined with the logical approach described in the earlier sections. Another approach to connecting goals and logics is considered in Section 6, in which a logical language is used to express requirements that are evaluated on a semantic level rather than with a logic for that language. Semantics are further tied to the earlier sections by assigning meanings to BAN constructs in the strand space model of authentication protocols [THG97, THG98b]. Finally, in Section 7 we say a few words about the future of formal analysis of authentication protocols.

2 BAN Logic

In this section we present an overview of BAN logic.[1] Specifically, we introduce the concepts, notation, and rules of BAN, after which, we will give some sample analyses. We will end the section by describing some of the extensions to BAN.

BAN is a logic of belief. The intended use of BAN is to analyze authentication protocols by deriving the beliefs that honest principals correctly executing a protocol can come to as a result of the protocol execution. For example, Alice might come to believe that a key she has received from a server is a good key for a communication session with Bob. What 'good' means here will be discussed below. The approach is to "idealize" the messages in the protocol specification into logical formulae. For example, if a server sends Alice a session key inside an encrypted message, the key might be replace by a formula that means that the key is good. We could then draw inferences based on Alice's ability to decrypt the key and other assumptions that might ultimately lead to the conclusion that she believes that the received key is good for talking with Bob.

BAN has been highly successful in uncovering protocol flaws, needed assumptions, etc., and it is relatively easy to use. A clear motivation in BAN is the math-

[1] 'BAN' is derived from the names of its authors, Burrows, Abadi and Needham. It is the first in a family of eponymous authentication logics. Versions of this logic occurred in many places. The first presentation in a public forum was at TARK in March of 1988 [BAN88]. It was also presented at the first CSFW in June of 1988. A revised and expanded version of the logic was given at SOSP in December of 1989 [BAN89b]. Journal versions of this appeared in *ACM TOCS* [BAN90a] and in *Proceedings of the Royal Society of London* [BAN89c]. The *TOCS* paper is an abbreviated version of the same material. The *Proc. Royal Society* paper is the one typically cited by the authors. Our primary source is the the Digital Systems Research Center report [BAN89a], and all descriptions of BAN are drawn from it. This SRC report originated in February 1989 and was revised in February 1990 to include "The Scope of a Logic of Authentication", which first appeared at a DIMACS workshop in October 1989 [BAN90c]. The February 1989 version of the SRC report comprises the *Proc. Royal Society* Paper.

ematician's credo to make only the needed distinctions and no more. Thus for example, the authors simplify reasoning about time by only distinguishing past and present epochs. These are not determined by the ephemeral current instant but are constant, set once and for all. This gives rise to one of BAN's central concepts, freshness. The other central concept is the aforementioned goodness of keys.

2.1 BAN Notation

We note at this point that the notation we will use is not that of [BAN89a]. Rather it is largely the notation introduced in [AT91] (with the public key notation introduced in [vO93]). It is closer to plain English than the original BAN notation, hence a bit more intuitive. For example, compare the following expressions (the first is original BAN notation):

$$P \models Q \mid\sim \#(X) \quad \text{vs.} \quad P \textit{ believes } Q \textit{ said } \textit{fresh}(X)$$

The language of BAN consists of the following expressions:

P believes X :
P received X : message; this may require decryption.
P said X :
P controls X :
fresh(X) : (Read 'X is fresh'.) X has not been sent in any message prior to the current protocol run.
$\boldsymbol{P \overset{k}{\longleftrightarrow} Q}$: (Read 'k is a good key for P and Q'.) k will never be discovered by any principal but P, Q, or a principal trusted by P or Q. (The last case is necessary, since the server often sees, indeed generates, k.)
PK(P, k) : (Read 'k is a public key of P'.) The secret key, k^{-1}, corresponding to k will never be discovered by any principal but P or a principal trusted by P.
$\boldsymbol{\{X\}_k}$: Short for "$\{X\}_k$ *from* P" (Read 'X encrypted with k (from P)'.) This is the notation for encryption. Principals can recognize their own messages. Encrypted messages are uniquely readable and verifiable as such by holders of the right keys.

In all of these expressions, "X" is either a message or a formula. As we will see, every formula can be a message, but not every message is a formula.

2.2 BAN Rules

In an analysis, the protocol is first idealized into messages containing assertions, then assumptions are stated, and finally conclusions are inferred based on the assertions in the idealized messages and those assumptions. The rules to do so are now given.

The first rule is called "message meaning". It and "nonce verification" are the central rules of BAN.

Message Meaning

$$\frac{P \textit{ believes } P \overset{k}{\longleftrightarrow} Q \qquad P \textit{ received } \{X\}_k}{P \textit{ believes } Q \textit{ said } X}$$

> "If P receives X encrypted with k and if P believes k is a good key for talking with Q, then P believes Q once said X." [BAN89a]

Note that this rule does not tell us anything about what submessage(s) P can extract from an encrypted message. That will come below under **Receiving Rules**. Rather, this rule tells us what P can discern about who sent the message. In applying symmetric keys, there is no explicit distinction between signing and encryption. (In BAN, there is also no distinction when applying public keys. Both signing and encryption are represented by $\{X\}_k$. The distinction is implicit in the notation for the key used: k or k^{-1}.)

There is also a public-key version of message meaning. The implied "*from*" field in the shared-key case would be redundant in the public-key case since it is implicit in the meaning of the notation for binding a public key to a principal who originated the signed message.

$$\frac{P \textit{ believes } PK(Q,k) \qquad P \textit{ received } \{X\}_{k^{-1}}}{P \textit{ believes } Q \textit{ said } X}$$

Nonce Verification

$$\frac{P \textit{ believes } \textit{fresh}(X) \qquad P \textit{ believes } Q \textit{ said } X}{P \textit{ believes } Q \textit{ believes } X}$$

This rule allows promotion from the past to the present (something said some time in the past to a present belief). In order to be applied, X should not contain any encrypted text. This rule is the only way for such promotion to occur. Since principals are all assumed to be honest and competent with respect to following the protocol, it makes sense that anything that a principal said recently should be something that he believes. That is, it makes sense for assertions, but what if X is a nonce, n? Obviously, it is a stretch of the intuitive meaning of belief to say that Bob believes a nonce. It is not necessary that this technical use respect all of our intuitions. The goal of the logic is to provide something useful for the analysis of authentication protocols, not to formalize reasoning about ordinary belief. Nonetheless, [BAN89a] suggests introducing a "has recently said" operator. This was in fact done by many later authors; although the motivation may have had more to do with making the belief part of the logic conform more closely to traditional modal logics of knowledge and belief.

Jurisdiction

$$\frac{P \textit{ believes } Q \textit{ controls } X \qquad P \textit{ believes } Q \textit{ believes } X}{P \textit{ believes } X}$$

The jurisdiction rule is what allows inferences that a principal believes a key is good, even though it is a random string that he has never seen before. An important thing to keep in mind about jurisdiction is the strength of *controls* statements. If P *controls* X, then P cannot make a mistake in asserting X. In BAN, this is somewhat tempered by only allowing inferences , e.g., from Alice's belief that the server controls $A \overset{k_{AB}}{\longleftrightarrow} B$ to Alice's belief that $A \overset{k_{AB}}{\longleftrightarrow} B$. In other words, inferences cannot be made about whether a key is actually good, but only about whether a key is believed to be good. Later logics, in particular AT and SVO, by separating off the axioms of belief, lose this tempering in principle. However, in practice this makes little difference.

Note that quantifiers are implicit. So "A *believes* S *controls* $A \overset{k}{\longleftrightarrow} B$" is implicit for "$A$ *believes* $\forall k.(S$ *controls* $A \overset{k}{\longleftrightarrow} B)$". Of course leaving things implicit leads in principle to some ambiguities. For example, "A *believes* $\forall k.(S$ *controls* B *controls* $A \overset{k}{\longleftrightarrow} B)$" vs. "$A$ *believes* S *controls* $\forall k.(B$ *controls* $A \overset{k}{\longleftrightarrow} B)$".

In practice, these questions do not usually arise in basic authentication protocols because nested assertions of jurisdiction are rarely found. As in other things, the BAN approach is to ignore complications not encountered in practice. If it should become necessary to be explicit, however, then they do so. For example, in [ABKL90], nested *controls* statements occur and the quantifiers in those statements are made explicit.

Belief Conjuncatenation

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

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

The obvious rules apply to beliefs concerning concatenations of messages/conjunctions of formulae. We have chosen the neologistic mouthful 'conjuncatenation' to again reinforce the point that BAN makes only the distinctions it needs. In this case, concatenations of messages are not distinguished from conjunctions of formulae: both are represented as (X, Y) in the above rules. Also, following the lead of [BAN89a], we do not list all of the rules; we give only a representative sampling. For example, we will not state versions of the last two rules where the conclusions are replaced by P *believes* Q *believes* Y and P *believes* Q *said* Y.

Freshness Conjuncatenation

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

For some inexplicable reason, this is a commonly misunderstood BAN rule. Some try to deny it; others try to assert the converse rule. Be wary of these mistakes. If X is fresh, then any message containing X is fresh in virtue of having X in it. But, (X, Y) being fresh tell us nothing about the freshness either of X by itself or of Y by itself (because the whole may be fresh in virtue of the other part).

Receiving Rules: Seeing is Receiving

$$\frac{P \text{ believes } P \overset{k}{\longleftrightarrow} Q \qquad P \text{ received } \{X\}_k}{P \text{ received } X} \qquad \frac{P \text{ received } (X, Y)}{P \text{ received } X}$$

A principal receiving a message also receives submessages he can uncover. Here is another clever BAN fusion, one that is lost a little in our more English-like notation: in BAN the symbol for receiving, '$\triangleleft$', is used to reason about what is visible. Thus, what a principal possesses is not distinguished from what he has received in some message. Virtually all successors to BAN distinguished the two; yet BAN is able to analyze a large number of protocols without this distinction.

This completes our listing of the rules of BAN. We now describe how to use BAN to analyze a protocol.

2.3 BAN Protocol Analysis

There are four steps to a protocol analysis using BAN.

1. Idealize the protocol.
2. Write assumptions about the initial state.
3. Annotate the protocol: For each message transmission "$P \rightarrow Q : \quad M$" in the protocol, assert Q *received* M.
4. Use the logic to derive the beliefs held by protocol principals.

As an example, we will go through a BAN analysis of the Needham-Schroeder Shared-Key protocol of Section 1.2.

Example: Analysis of Needham-Schroeder Shared-Key (NSSK)

The first step is to put the protocol into idealized form.

Idealized Needham-Schroeder Shared-Key [BAN89a]

Message 2 $S \rightarrow A : \{n_A, A \overset{k_{AB}}{\longleftrightarrow} B, \mathit{fresh}(k_{AB}), \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}}\}_{k_{AS}}$ *from* S

Message 3 $A \rightarrow B : \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}}$ *from* S

Message 4 $B \rightarrow A : \{n_B, A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{AB}}$ *from* B

Message 5 $A \rightarrow B : \{n_B, A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{AB}}$ *from* A

Note that the first message of the protocol is omitted in the idealized form. In a BAN idealization, plaintext from the protocol is omitted. Next note the *from* fields. It is always assumed that principals can recognize their own messages. Thus, with a shared key, if a recipient can decrypt a message, she can tell who it is from. As this is often implicitly clear, the *from* field is often omitted from the protocol idealization. What is inside the encrypted messages is also altered. Specifically, the key k_{AB} is replaced by assertions about it. So, Message 2, idealized, is an encrypted message for Alice from the Server that contains a nonce, an assertion that k_{AB} is fresh, an assertion that k_{AB} is good for talking to Bob, and an encrypted message to be forwarded to Bob. Note also that in the last message $n_B - 1$ is changed to just n_B. This is because the purpose of subtracting 1 from the nonce is to differentiate it from Message 4. The differentiation is reflected in the idealization in the *from* field. It would reduce informal interpretation to simply leave $n_B - 1$ in the idealized protocol. But, BAN has no direct rule to infer the freshness of $n_B - 1$ from the freshness of n_B. So, the change is necessary. This was changed in SVO, as we shall see below.

Once the idealization has been made, assumptions are stated.

NSSK Initial State Assumptions

P1. A *believes* $A \overset{k_{AS}}{\longleftrightarrow} S$
P2. B *believes* $B \overset{k_{BS}}{\longleftrightarrow} S$

P3. A *believes* S *controls* $A \overset{k}{\longleftrightarrow} B$
P4. B *believes* S *controls* $A \overset{k}{\longleftrightarrow} B$
P5. A *believes* S *controls* $fresh(A \overset{k}{\longleftrightarrow} B)$

P6. A *believes* $fresh(n_A)$
P7. B *believes* $fresh(n_B)$

Most of the assumptions should be self-explanatory. P1 and P2 express the belief in the quality of the long term keys. (Notice that we make no corresponding assumptions about what S believes. It would be natural to do so, but we have omitted them because, in this case, they are not needed in the derivations that follow.) P3 through P5 give the assertions on which Alice and Bob believe that the server has jurisdiction. P6 and P7 tell us that each principal believes that his/her random value is fresh.

NSSK Annotated Protocol

The annotation states assumptions based on the messages in the idealized protocol. It can be read directly from the idealization.

P8. A *received* $\{n_A, A \overset{k_{AB}}{\longleftrightarrow} B, fresh(k_{AB}), \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}}\}_{k_{AS}}$
P9. B *received* $\{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}}$ *from* S
P10. A *received* $\{n_B, A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{AB}}$ *from* B
P11. B *received* $\{n_B - 1, A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{AB}}$ *from* A

This completes the assumptions needed to analyze the protocol. In the derivations below, every line is followed by a justification, i.e., the rule by which it was derived and the premise(s) and/or derived formula(e) used in its derivation.

NSSK Derivations

1. A *believes* S *said* $(n_A, A \overset{k_{AB}}{\longleftrightarrow} B, \mathit{fresh}(A \overset{k_{AB}}{\longleftrightarrow} B), \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}})$
   By Message Meaning using P1, P8.
2. A *believes* $\mathit{fresh}(n_A, A \overset{k_{AB}}{\longleftrightarrow} B, \mathit{fresh}(A \overset{k_{AB}}{\longleftrightarrow} B), \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}})$
   By Freshness Conjuncatenation using 1, P6.
3. A *believes* S *believes* $(n_A, A \overset{k_{AB}}{\longleftrightarrow} B, \mathit{fresh}(A \overset{k_{AB}}{\longleftrightarrow} B), \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{BS}})$
   By Nonce Verification using 2, 1.
4. A *believes* S *believes* $(A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Belief Conjuncatenation using 3.
5. A *believes* S *believes* $(\mathit{fresh}A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Belief Conjuncatenation using 3.
6. A *believes* $(A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Jurisdiction using 4, P3.
7. A *believes* $\mathit{fresh}(A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Jurisdiction using 4, P5.

We have derived Alice's belief in the goodness and in the freshness of k_{AB}. We now turn to Bob.

8. B *believes* S *said* $A \overset{k_{AB}}{\longleftrightarrow} B$
   By Message Meaning using P2, P9.

With the assumptions we have made so far this is all we are able to derive with respect to Bob's belief in the goodness of k_{AB}. Unlike Alice, Bob has sent no nonce at this point in the protocol. The only way for us to move further is if we assume that Bob believes something he has received is fresh. We therefore add the assumption that Bob believes that the assertion $A \overset{k_{AB}}{\longleftrightarrow} B$ is fresh.

P12. B *believes* $\mathit{fresh}(A \overset{k_{AB}}{\longleftrightarrow} B)$

This is different than our earlier freshness assumptions since those were all based on values that the believing principal had herself generated. This one expresses Bob's belief that a random value someone else has generated is fresh. We will return to this odd assumption below after we complete the derivations.

9. B *believes* S *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
   By Nonce Verification using P12, 8.
10. B *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
    By Jurisdiction using P4, 9.

Unlike for A, for B we were forced to assume B *believes* $\mathit{fresh}(A \overset{k_{AB}}{\longleftrightarrow} B)$ since we were unable to derive it. We have now derived Alice and Bob's first order beliefs in the goodness and freshness of k_{AB}. We next derive their second order beliefs.

11. *A believes B said* $(n_B, A \overset{k_{AB}}{\longleftrightarrow} B)$
    By Message Meaning using 6, P10.
12. *A believes fresh*$(n_B, A \overset{k_{AB}}{\longleftrightarrow} B)$
    By Freshness Conjuncatenation using 7.
13. *A believes B believes* $(n_B, A \overset{k_{AB}}{\longleftrightarrow} B)$
    By Nonce Verification using 12, 11.
14. *A believes B believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
    By Belief Conjuncatenation using 13.

By similar reasoning, we can obtain *B believes A believes* $A \overset{k_{AB}}{\longleftrightarrow} B$ but with an important difference. Since Bob believes that $A \overset{k_{AB}}{\longleftrightarrow} B$ is fresh, there is no need for n_B in order for him to reach this conclusion. The only role n_B plays in the protocol is to differentiate Message 4 from Message 5. It does not need to be fresh for that. This makes the assumption that Bob believes $A \overset{k_{AB}}{\longleftrightarrow} B$ to be fresh stand out all the more. We now illustrate that the dubious nature of this assumption is not just an artifact of the analysis.

The Denning-Sacco Attack

In 1981, Denning and Sacco showed how the Needham-Schroeder Shared-Key protocol could be attacked if an attacker compromised an old session key [DS81]. In the attack specification 'E_A' is the attacker masquerading as A.

Message 3 $E_A \rightarrow B : \{k_{AB}, A\}_{k_{BS}}$
Message 4 $B \rightarrow E_A : \{n'_B\}_{k_{AB}}$
Message 5 $E_A \rightarrow B : \{n'_B - 1\}_{k_{AB}}$

The attack relies on the fact that Bob has no way to actually be assured that Message 3 is fresh. So, an attacker could spend whatever time is needed to break the session key k_{AB}. As long as she can do so within the lifetime of k_{BS}, then she can run the above attack. Bob will then think he has confirmed sharing k_{AB} with Alice, when in reality Alice is not present and the attacker knows the key. The attack is not directly uncovered by a BAN analysis of the protocol; rather the analysis shows that the protocol cannot achieve any sort of authentication for Bob without making the dubious assumption that underlies the attack.

This concludes our basic introduction to BAN logic and its use in protocol analysis. In the remainder of this section, we will set out some of the issues that led researchers to expand and modify BAN and the analysis technique.

2.4 The Nessett Protocol

In 1990, Nessett introduced the following simple example that "demonstrates that a significant flaw exists in the Burrows, Abadi and Needham logic" [Nes90].

Protocol 2 (Nessett) [Nes90]

Message 1 $A \rightarrow B: \quad \{n_A, k_{AB}\}_{k_A^{-1}}$

Message 2 $B \rightarrow A: \quad \{n_B\}_{k_{AB}}$

In the first message, Alice encrypts a session key using a private key, the public cognate of which is assumed to be widely known. Bob then send her a handshake value encrypted with the session key, k_{AB}. Of course the key is not at all good for communication between Alice and Bob because anyone can extract it from the first message and use it for communication. But, structurally the protocol is the same as one where the first message was encrypted with a good key. Here is Nessett's idealization of the protocol followed by the corresponding annotation, then by the initial state assumptions Nessett presents.

Idealized Nessett Protocol

Message 1 $A \rightarrow B: \quad \{n_A, A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_A^{-1}}$

Message 2 $B \rightarrow A: \quad \{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{AB}}$

Annotation Premises

P1. B *received* $\{n_A, A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_A^{-1}}$

P2. A *received* $\{A \overset{k_{AB}}{\longleftrightarrow} B\}_{k_{AB}}$

Initial State Assumptions

P3. B *believes* $PK(k_A, A)$

P4. A *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$

P5. A *believes* $fresh(A \overset{k_{AB}}{\longleftrightarrow} B)$

P6. B *believes* $fresh(n_A)$

P7. B *believes* A *controls* $(A \overset{k}{\longleftrightarrow} B)$

Note that Nessett assumes Bob to believe that n_A is fresh. Therefore, it is more naturally thought of as a timestamp than a nonce. We have used this notation to more closely follow Nessett[2]. Based on this idealization and set of assumptions, Nessett is makes the following derivations.

Nessett Protocol Derivations

1. B *believes* A *said* $(n_A, A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Message Meaning using P3, P1.

[2] Specifically, he used "N_A".

2. *B believes fresh*$(n_A, A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Freshness Conjuncatenation using P6.
3. *B believes A believes* $(n_A, A \overset{k_{AB}}{\longleftrightarrow} B)$
   By Nonce Verification using 2, 1.
4. *B believes A believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
   By Belief Conjuncatenation using 3.
5. *B believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
   By Jurisdiction using P7, 4.

This completes the derivations for Bob. We now derive Alice's second order belief in the goodness of k_{AB}. (Her first order belief was assumed.)

6. *A believes B said* $A \overset{k_{AB}}{\longleftrightarrow} B$
   By Message Meaning using P4, P2.
7. *A believes B believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
   By Nonce Verification using P5, 6.

Nessett's Critique of BAN

Using BAN, one can derive all of the typical BAN authentication goals for both Alice and Bob via the Nessett protocol—as we have just done. This shows that, according to BAN, k_{AB} is a good session key. But, k_{AB} is not a good key. Ergo, BAN is flawed. Nessett traces the source of the "flaw" to the scope of BAN. It addresses who gets and acknowledges a key (authentication), but it does not address who should not get a key (confidentiality).

Burrows et al. respond to Nessett in [BAN90b] by noting that their paper explicitly limits discussion to authentication of honest principals. They explicitly do no attempt to detect unauthorized release of secrets. Since Alice publishes k_{AB} in first message, Nessett's assumption *A believes* $A \overset{k_{AB}}{\longleftrightarrow} B$ is inconsistent with these stated restrictions. And, from absurd assumptions come absurd conclusions. The logic does not preclude ridiculous assumptions.

> "This seems fair enough; no logic protects against the assumption of bad premises. All one can reasonably ask is that, if the premises are true, then the conclusion is also true. On the other hand, Nessett could counter that illustrative counterexamples are supposed to be obvious; that's what makes them illustrative. If one wants to demonstrate that an argument form is invalid, one constructs an argument with that form that is clearly invalid. It is hardly fair to say that this shows nothing because it's an obviously invalid argument. That's the point. The danger is that one might use the form to justify an invalid argument that is not obviously so. The question remains: On what (extralogical) basis do we decide what goes into the premise set? For this case, Burrows et al. would no doubt contend that one includes a principal's belief in the goodness of a key only if she has reason to believe that it is good and no reason to think that it is not. Of course that is what we would like to do, but it is also what we are trying to determine how to do." [Syv91]

Certainly it is much too strong to say that the Nessett example shows the logic to be flawed. It does highlight a place where one is expected to rely purely on the intuitive reasonableness of assumptions. However, it has not shown that this results in either a logical error or a practical vulnerability.

Still, it would be nice to have a way to capture either formally or at least rigorously, the difference between Nessett-type protocols and those not flawed in this way. Alice's action is inconsistent with the meaning of A *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$. What is needed is a way to reflect this mathematically [Syv91, Syv92]. Suppose we could derive A *believes* C *has* k_{AB} (for arbitrary C). Increasing expressiveness would let us formally demonstrate this.

2.5 Expanding Beyond BAN

In 1990, Gong, Needham, and Yahalom, introduced a new logic [GNY90] that extended BAN. This logic came to be known as *GNY*, following the precedent set by 'BAN'. In it one can represent possession of keys. So A *believes* C *has* k_{AB} can be expressed, and possibly derived. GNY also distinguishes available messages from received messages. Other important contributions include formalizing a principal's distinction of his own generated messages from others. Analysis in GNY also leaves cleartext in idealized protocols, rather than assuming that it cannot play a role in authentication. While not specifically formulated as a response to Nessett, this logic allows the expression of key possession and thus can express formally that with which the dubious assumption is supposed to be inconsistent. By itself this does not guarantee that the needed key possession is derivable, nor does it directly express the inconsistency.

Another response to Nessett that comes closer to directly reflecting the inconsistency of meaning was first given by Abadi and Tuttle in [AT91]. Specifically, they presented a BAN-like logic that possessed an independently motivated account of meaning in the form of a model-theoretic semantics. This allows one to rigorously assess the truth of assumptions (consistent with a protocol). Specifically, the *AT* logic was closer to traditional modal logics than BAN, provided a detailed model of computation, had a soundness result with respect to the model, and was also more expressive (e.g., could express key possession). A traditional semantics was much of the motivation for this work. While the published soundness result had a mistaken assumption in it, this was a large step towards putting BAN on a footing both firmer and logically more traditional. We will return to semantics below.

Another important limitation on BAN is the type of protocols to which it can be applied. Diffie-Hellman protocols underly much of modern authenticated key distribution. For example, they underly the IETF standard Internet Key Exchange (IKE) protocol [DH99], as well as both SSL and TLS—what your Web browser uses when it makes a secure connection. Thus, being able to reason about such protocols would be quite useful. Paul van Oorschot's VO logic [vO93] was designed primarily to add this capability. It is an extension of GNY that can be used to reason about Diffie-Hellman type key agreement. In addition, [vO93]

extended the lexicon of formally stated authentication properties, and formalized reasoning about confirmed possession of secrets. We will return to those points in Section 4. Right now we turn to a logic that attempts to comprise all of the advantages that VO and the other BAN logics introduce.

3 SVO Logic: Unifying the Predecessors

In the last section, we saw some of the limitations of BAN, and the extensions and variants that were intended to overcome them. Each of the logics, GNY, AT, and VO brought a distinct addition. In response to this diversity, Syverson and van Oorschot devised a logic, SVO, that was intended to unify the above predecessors [SvO94, SvO96]. The intent was not simply to patch on new notation and rules adequately expressive to capture the additional scope of these logics. This would be both inelegant and potentially unsound. Rather, the intent was to produce a model of computation and a logic that was sound with respect to that model while still retaining the expressiveness of the various BAN extensions.

3.1 SVO Notation

SVO uses the notation already introduced for BAN, with the following main additions:

$\neg\varphi$: Negations of formulae are added to the language.
$\boldsymbol{P}$ ***says*** $\boldsymbol{X}$: but P must have said X since the beginning of current epoch.
$\boldsymbol{P}$ ***has*** $\boldsymbol{X}$:

- Initially available to P,
- Received by P,
- Freshly generated by P, and
- Constructible by P from the above.

The original BAN idea of a public key might be expressed as '$PK(P,k)$', meaning that k is a public key of P — the matching secret key k^{-1} will never be discovered by any principal but P or a principal trusted by P. In SVO, this is refined to cover different types of public key functionality.

$\boldsymbol{PK_{\psi}(P,k)}$: k is a public ciphering key of P. Only P can read messages encrypted with k.
$\boldsymbol{PK_{\sigma}(P,k)}$: k is a public signature key of P. The key k verifies that messages signed by k^{-1} are from P.
$\boldsymbol{PK_{\delta}(P,k)}$: k is a public key-agreement key of P. A Diffie-Hellman key formed with k is shared with P.
$\boldsymbol{\lfloor X \rfloor_k}$: types of public keys, SVO distinguishes signature from encryption.
$\boldsymbol{\{X\}_k}$: This is no longer short for '$\{X\}_k$ *from* P'. In SVO, it is not assumed that principals can recognize their own messages. But it is still assumed that encrypted messages are uniquely readable and verifiable as such by holders of the right keys.

$\langle \boldsymbol{X} \rangle_{*\boldsymbol{P}}$: of [WK96] and is closer to that than to the notation in [SvO94, SvO96]. This is used for messages that P doesn't know or recognize (e.g., $\{X\}_k$ where P does not know k). P will nonetheless recognize $\langle \{X\}_k \rangle_{*P}$ as the same thing if received again, even if he cannot tie it back to any plaintext.

X from P :

3.2 SVO Rules

Like AT, SVO is much more of a traditional axiomatic style logic than BAN, GNY or VO. As such there are only two rules.

Modus Ponens

$$\frac{\varphi \qquad \varphi \rightarrow \psi}{\psi}$$

Necessitation

$$\frac{\vdash \varphi}{\vdash P \textit{ believes } \varphi}$$

'$\vdash$' is a metalinguistic symbol. '$\Gamma \vdash \varphi$' means that the formula φ is derivable from the set of formulae Γ (and the axioms as stated below) using the above rules. '$\vdash \varphi$' is short for '$\emptyset \vdash \varphi$' and means that φ is a theorem, i.e., derivable from axioms alone without any additional assumptions. We describe derivability (i.e. proofs) below in Section 3.4.

Necessitation is sometimes called by other names, e.g., belief generalization. We have given it the name that reflects its origins in alethic modal logic [Che80]. It applies only to theorems of the logic. Like the Belief Conjuncatenation rule of BAN, this is often misunderstood and misapplied or improperly criticized. If using the assumptions about a protocol it is possible to derive that Q *said* X, it does not follow that P *believes* Q *said* X. This is because Q *said* X is merely derivable given the context of this protocol: it is not a theorem, i.e., derivable using logic alone.

Axioms of the logic are all instances of tautologies of classical propositional calculus [Men87], and all instances of the following axiom schemata.[3]

3.3 SVO Axioms

Belief Axioms

1. $(P \textit{ believes } \varphi \wedge P \textit{ believes } (\varphi \rightarrow \psi)) \rightarrow P \textit{ believes } \psi$
2. $P \textit{ believes } \varphi \rightarrow \varphi$
3. $P \textit{ believes } \varphi \rightarrow P \textit{ believes } (P \textit{ believes } \varphi)$
4. $\neg(P \textit{ believes } \varphi) \rightarrow P \textit{ believes } (\neg P \textit{ believes } \varphi)$

[3] 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'.

These are classic axioms of modal logic [Che80, Gol92] that were thoroughly analyzed in the early and middle part of the last century. (The classic names for them are **K**, **T**, **4**, and **5** respectively.) As in AT, belief is removed from most other axioms, the logic of belief is separated off from the rest of the logic. Readers familiar with modal logic will recognize these as the axioms[4] of the Lewis system **S5** [Che80]. This logical system is usually taken to characterize knowledge rather than belief, so we provide a brief side discussion of the point. It may be skipped without loss of continuity.

Discussion: Knowledge and Belief

Roger Needham has remarked in conversation that perhaps the biggest mistake they made with BAN was calling it a belief logic. We have already seen in the discussion of Nonce Verification in Section 2.2 that BAN-belief does not respect all of the intuitions of ordinary belief. Of course no technical usage could. Intuitions can be helpful, but they can also lead us from the main task of providing a useful analysis of authentication and authentication protocols. More detailed discussion of knowledge and belief in authentication logics can be found in [Syv92, AT91]. Here we cover a few of the main points.

The main question in distinguishing knowledge from belief is: If a principal thinks that φ, is he always right? If the answer is "Yes", then we are talking about knowledge. If the answer is "No", then we are talking about belief. That is overly simplistic, and philosophical counterexamples are thoroughly plumbed in the literature (e.g., look up the Gettier Problem [Mos86]). Nonetheless, it will do for our practical intentions. This translates naturally into the main formal question: Is it a theorem of the logic that (P *believes* $\varphi \rightarrow \varphi$)?

This is just the **T** axiom introduced above as axiom 2. Faced now with this technical distinction, we can ask the practical question of which do we want. Surprisingly, the answer for our purposes is "I don't care." The reason is that this has played little role in actual analyses. So, if we don't care, how did we come to add it as an axiom?

The answer lies in the semantics of the logic. One of the goals of this logic was that it have an intuitively reasonable model of computation and semantics, and that it be sound with respect to them. Mark Tuttle has noted that we ought to build models not logics in trying to capture our notions of authenticated communication [Tut]. It turns out that in [SvO96], the semantics of the intentional operator is based on an equivalence relation. And, it is a classic result of modal logic, that this is characteristic of the logic **S5**. So, it simply falls out that the above axioms are all valid in our semantics. But, this is not an important practical distinction. In practice, only axioms 1 and 3 seem to play a role.

Source Association Axioms

5. $(P \xleftrightarrow{k} Q \wedge R \textit{ received } \{X \textit{ from } Q\}_k) \rightarrow (Q \textit{ said } X \wedge Q \textit{ has } X)$

[4] The **4** axiom (our axiom 3) is actually derivable given the others and is included for tradition and for its intuitive significance.

This replaces the Message Meaning Rule of BAN. Note the absence of the '*believes*' operator, and that the axiom applies when any principal R receives $\{X \textit{ from } Q\}_k$. The logical significance of '$P \overset{k}{\longleftrightarrow} Q$' is now isolated from that of *believes*. As in BAN, there is also a corresponding public-key axiom.

6. $(PK_\sigma(Q,k) \wedge R \textit{ received } X \wedge SV(X,k,Y)) \rightarrow Q \textit{ said } Y$

We introduce some new notation here. '$SV(X,k,Y)$' means that applying k to X confirms that X is the result of signing Y with a private cognate of k. Note that this axiom separates out key binding (what principal is bound to the key) from key correctness (what key verifies the signature).

Key Agreement Axioms

7. $(PK_\delta(P,k_P) \wedge PK_\delta(Q,k_Q)) \rightarrow P \overset{F_0(k_P,k_Q)}{\longleftrightarrow} Q$
8. $\varphi \equiv \varphi[F_0(k,k')/F_0(k',k)]$ $F_0(k',k)$ implicitly names the (Diffie-Hellman) function that combines k' with k^{-1} to form a shared key.

These are the axioms that characterize Diffie-Hellman key agreement. As we mentioned in Section 2.5, this is an important component in widely used authenticated key establishment protocols. We give here a brief account of the mathematics of Diffie-Hellman. For more details consult a standard cryptography text [MvOV97, Sch96].

Protocol 3 (Diffie-Hellman)

1. Assume that Alice and Bob share
   - a large prime p
   - a generator g of the multiplicative group $\mathbb{Z}_p^*$ of integers *modulo* p.
2. Alice chooses large integer x and computes $X = g^x \bmod p$
3. Bob chooses large integer y and computes $Y = g^y \bmod p$
4. Message 1 $A \rightarrow B: \; X$
   Message 2 $B \rightarrow A: \; Y$
5. Alice sets $k_{AB} = X^y \bmod p = g^{xy} \bmod p \;\; (= g^{yx} \bmod p)$
6. Bob sets $k_{AB} = Y^x \bmod p = g^{yx} \bmod p \;\; (= g^{xy} \bmod p)$

Receiving Axioms

9. $P \textit{ received } (X_1, \ldots X_n) \rightarrow P \textit{ received } X_i$, for $i = 1, \ldots, n$
10. $(P \textit{ received } \{X\}_{k^+} \wedge P \textit{ has } k^-) \rightarrow P \textit{ received } X$
    Here k^+ and $k-$ are used to abstractly represent cognate keys, whether for symmetric or asymmetric cryptography. In the symmetric case, $k^+ = k^- = k$. In the asymmetric case, k^+ is a public key and k^- the associated private key.
11. $(P \textit{ received } \lfloor X \rfloor_k) \rightarrow P \textit{ received } X$
    Principals are assumed to possess public keys (for convenience).

Possession Axioms

12. P *received* $X \rightarrow P$ *has* X
13. P *has* $(X_1, \ldots, X_n) \rightarrow P$ *has* X_i, for $i = 1, \ldots, n$.
14. $(P$ *has* $X_1 \wedge \ldots \wedge P$ *has* $X_n) \rightarrow P$ *has* $F(X_1, \ldots, X_n)$
    'F' is meta-notation for any function computable in practice by P, e.g., encryption with known keys. The meaning of "computable in practice" is intentionally not formally determined. It could be, e.g., polynomial-time computable but will be treated as a black box, just as "encryption", "signature", etc., are in nearly all formal treatments of cryptographic protocols.

Comprehension Axiom

15. P *believes* $(P$ *has* $F(X)) \rightarrow P$ *believes* $(P$ *has* $X)$
    'F' is meta-notation for any function that is effectively one-one (e.g., collision free hashes) and such that F^+ or F^- is computable in practice by P. Note that this axiom does not imply that F is invertible by P. Note also that the converse of this axiom is a derivable theorem.

Saying Axioms

16. P *said* $(X_1, \ldots, X_n) \rightarrow P$ *said* $X_i \wedge P$ *has* X_i, for $i = 1, \ldots, n$.
17. P *says* $(X_1, \ldots, X_n) \rightarrow (P$ *said* $(X_1, \ldots, X_n) \wedge P$ *says* $X_i)$, for $i = 1, \ldots, n$.

Freshness Axioms

18. *fresh*$(X_i) \rightarrow$ *fresh*$(X_1, \ldots, X_n)$, for $i = 1, \ldots, n$.
19. *fresh*$(X_1, \ldots, X_n) \rightarrow$ *fresh* $F(X_1, \ldots, X_n)$ F must genuinely depend on all component arguments. This means that it is infeasible to compute value of F without value of all the X_i.

Jurisdiction and Nonce-Verification Axioms

20. $(P$ *controls* $\varphi \wedge P$ *says* $\varphi) \rightarrow \varphi$
21. (*fresh*$(X) \wedge P$ *said* $X) \rightarrow P$ *says* X

Note: Neither axiom refers to belief

Symmetric Goodness Axiom

22. $P \overset{k}{\longleftrightarrow} Q \equiv Q \overset{k}{\longleftrightarrow} P$

3.4 Protocol Analysis

We now demonstrate how to do a protocol analysis using SVO. We will continue with our running example of the Needham-Schroeder Shared-Key Protocol 1. A comparison between BAN and SVO analysis is summarized in Figure 1.

BAN Analysis	**SVO Analysis**
1. Idealize the protocol.	
2. Write assumptions about initial state.	a. Write assumptions about initial state.
3. Annotate protocol. For each message "$P \to Q : M$" of the idealized protocol, assert "*Q received M*".	b. Annotate protocol. For each message "$P \to Q : M$" of the (not idealized) protocol, assert "*Q received M*".
	c. Assert comprehensions of received messages.
	d. Assert interpretations of comprehended messages.
4. Use the logic to derive the beliefs held by protocol principals.	e. Use the logic to derive beliefs held by protocol principals.

Fig. 1. Protocol analysis steps

NSSK Initial State Assumptions

As with BAN, the first assumptions to set out are the initial state assumptions. Unlike BAN, we do not idealize the protocol first. The assumptions are virtually the same as the initial state assumptions set out in the above BAN analysis, except that the jurisdiction assumption P5 is about jurisdiction over freshness of keys rather than jurisdiction over freshness of assertions about the goodness of keys.

P1. A *believes* $A \overset{k_{AS}}{\longleftrightarrow} S$
P2. B *believes* $B \overset{k_{BS}}{\longleftrightarrow} S$

P3. A *believes* S *controls* $A \overset{k}{\longleftrightarrow} B$
P4. B *believes* S *controls* $A \overset{k}{\longleftrightarrow} B$
P5. A *believes* S *controls* $fresh(k)$

P6. A *believes* $fresh(n_A)$
P7. B *believes* $fresh(n_B)$

P8. B *believes* $fresh(A \overset{k_{AB}}{\longleftrightarrow} B)$

NSSK Received Message Assumptions

This step is the same as the annotation assumptions in BAN, except that we here use the specified protocol, not its idealization. Amongst other things, this means that plaintext is not eliminated, and these premises can be read directly from the specification. These premises are not typically used directly in derivations. Rather, they are used in the production of comprehension premises, which are themselves used in producing interpretation premises.

P9. S *received* (A, B, n_A)
P10. A *received* $\{n_A, B, k_{AB}, \{k_{AB}, A\}_{k_{BS}}\}_{k_{AS}}$
P11. B *received* $\{k_{AB}, A\}_{k_{BS}}$
P12. A *received* $\{n_B\}_{k_{AB}}$
P13. B *received* $\{n_B - 1\}_{k_{AB}}$

NSSK Comprehension Assumptions

In this step, we express that which a principal comprehends of a received message. The move from the received message assumptions is usually straightforward in practice. In principle, this can be formalized. But a rigorous formalization makes for a very complicated logic and some of the intuitiveness of SVO is lost. Nonetheless, it may be desirable if the intent is to automate as much of the reasoning as possible. (Instances of such a formalization can be found in [WK96, Dek00].)

P14. S *believes* S *received* $(A, B, \langle n_A \rangle_{*s})$
P15. A *believes* A *received* $\{n_A, B, \langle k_{AB} \rangle_{*A}, \langle \{k_{AB}, A\}_{k_{BS}} \rangle_{*A}\}_{k_{AS}}$
P16. B *believes* B *received* $\{\langle k_{AB} \rangle_{*B}, A\}_{k_{BS}}$
P17. A *believes* A *received* $\{\langle n_B \rangle_{*A}$ *from* $B\}_{\langle k_{AB} \rangle_{*A}}$
P18. B *believes* B *received* $\{n_B - 1\}_{\langle k_{AB} \rangle_{*B}}$

NSSK Interpretation Assumptions

These assumptions are essentially the replacement for idealization. Producing them is inherently an informal process. We are asserting how a principal interprets a received message (as that principal understands it). This is inherently dependent on the protocol design. Idealization is one of the most criticized and/or misapplied aspects of BAN analysis—bad initial state assumptions being the other. While some informality seems necessary in anything like this framework, SVO analysis reduces the potential for problems. First, idealization is split into comprehension and interpretation. Second, and perhaps more important, the interpretational part of the process occurs after annotation rather than before. In idealization, there is a natural and correct tendency to interpret message components using formulae expressing the intent of the sender. BAN annotation then asserts that the receiver receives the intended meaning of the sender. By placing interpretation after annotation and comprehension, the focus naturally shifts to how the intent of the sender is understood by the receiver. That is, focus shifts from the meaning the sender had intended to the meaning that the receiver attaches to a received message.

P19. A *believes* A *received* $\{n_A, B, \langle k_{AB} \rangle_{*A}, \langle \{k_{AB}, A\}_{k_{BS}} \rangle_{*A}\}_{k_{AS}} \rightarrow A$ *believes* A *received* $\{n_A, B, A \overset{\langle k_{AB} \rangle_{*A}}{\longleftrightarrow} B, \mathit{fresh}(\langle k_{AB} \rangle_{*A}), \langle \{k_{AB}, A\}_{k_{BS}} \rangle_{*A}\}_{k_{AS}}$
P20. B *believes* B *received* $\{\langle k_{AB} \rangle_{*B}, A\}_{k_{BS}} \rightarrow$
B *believes* B *received* $\{A \overset{\langle k_{AB} \rangle_{*B}}{\longleftrightarrow} B, \mathit{fresh}(\langle k_{AB} \rangle_{*B}), A\}_{k_{BS}}$
P21. (A *believes* A *received* $\{\langle n_B \rangle_{*A}\}_{\langle k_{AB} \rangle_{*A}}) \wedge (A$ *believes* $A \overset{\langle k_{AB} \rangle_{*A}}{\longleftrightarrow} B) \rightarrow$
A *believes* A *received* $\{\langle n_B \rangle_{*A}, A \overset{\langle k_{AB} \rangle_{*A}}{\longleftrightarrow} B\}_{\langle k_{AB} \rangle_{*A}}$
P22. (B *believes* B *received* $\{n_B - 1\}_{\langle k_{AB} \rangle_{*B}} \wedge (B$ *believes* $A \overset{\langle k_{AB} \rangle_{*B}}{\longleftrightarrow} B) \rightarrow$
B *believes* B *received* $\{n_B - 1, A \overset{\langle k_{AB} \rangle_{*B}}{\longleftrightarrow} B\}_{\langle k_{AB} \rangle_{*B}}$

Another point to note about these premises is that they all have conditional form. Often, the conditional is only a formal reminder that the interpretation depends on the comprehension of the actual receipt of the message. But in some cases, e.g., assumption P21, the interpretation depends not just on receipt of a message but also on other things, such as assumptions about good keys.

We have omitted an interpretation premise for the first message because it will play no role in the derivations. (The BAN assumption that plaintext does not affect analysis is not merely capricious.)

NSSK Derivations for Alice

1. A *believes* A *received* $\{n_A, B, A \xleftrightarrow{\langle k_{AB}\rangle_{*A}} B, \mathit{fresh}(\langle k_{AB}\rangle_{*A}), \langle\{k_{AB}, A\}_{k_{BS}}\rangle_{*A}\}_{k_{AS}}$
   By Modus Ponens using P19, P15

Unlike in BAN, to move from here to Alice believing that the server sent the message she received requires several steps. We would have to apply the necessitation rule to the source association axiom, and also make use of propositional reasoning, axiom 1, modus ponens, etc. We will present here only the highlights, focusing on the authentication reasoning. We will generally omit reference to the rules (modus ponens and necessitation) and will refer to propositional reasoning or reasoning using the belief axioms by saying "by Belief Axioms".

2. A *believes* S *said* $\{n_A, B, A \xleftrightarrow{\langle k_{AB}\rangle_{*A}} B, \mathit{fresh}(\langle k_{AB}\rangle_{*A}), \langle\{k_{AB}, A\}_{k_{BS}}\rangle_{*A}\}_{k_{AS}}$
   By Source Association, 1, P1, and Belief Axioms
3. A *believes* S *says* $\{n_A, B, A \xleftrightarrow{\langle k_{AB}\rangle_{*A}} B, \mathit{fresh}(\langle k_{AB}\rangle_{*A}), \langle\{k_{AB}, A\}_{k_{BS}}\rangle_{*A}\}_{k_{AS}}$
   By Freshness, Nonce Verification, 2, P6, and Belief Axioms
4. A *believes* $A \xleftrightarrow{\langle k_{AB}\rangle_{*A}} B$
   By Saying, Jurisdiction, 3, P3, and Belief Axioms
5. A *believes* $\mathit{fresh}(\langle k_{AB}\rangle_{*A})$
   By Saying, Jurisdiction, 3, P5, and Belief Axioms
6. A *believes* B *said* $(\langle n_B\rangle_{*A}, A \xleftrightarrow{\langle k_{AB}\rangle_{*A}} B)$
   By Source Association, P21, 4, and Belief Axioms
7. A *believes* B *has* $\langle k_{AB}\rangle_{*A}$
   By Source Association, P21, 4, and Belief Axioms
8. A *believes* B *says* $(\langle n_B\rangle_{*A}, A \xleftrightarrow{\langle k_{AB}\rangle_{*A}} B)$
   By Freshness, Nonce Verification, 5, 6, and Belief Axioms

We could obtain similar results for Bob (assuming we use the dubious assumption P8, the dubiousity of which is discussed in Section 2.3). This concludes our analysis of the Needham-Schroeder Shared-Key protocol. As noted above, one of the motivations of SVO was to incorporate reasoning about Diffie-Hellman style key agreement. Analyses of two such protocols can be found in [SvO96].

3.5 The Nessett Protocol in SVO

What about the Nessett Protocol? How does it fare in SVO? Since SVO contains both negation and the ability to express possession, it can express who should not get keys. This was Nessett's primary concern about BAN.

More precisely. In SVO, we can state the requirement

$$\neg(E \textit{ has } k_{AB})$$

where 'E' is the adversary. Now, given our assumption of a Dolev-Yao adversary, it is perfectly reasonable, for every message M of every protocol to add to the annotation assumptions that E *received* M. It then becomes trivial for the Nessett protocol to derive

$$E \textit{ has } k_{AB}$$

So, we can prove Nessett Protocol to be insecure. But, what if we could not prove E *has* k_{AB}? What if this were merely consistent with the protocol not provable from it? As has been observed, failed proofs sometimes reveal attacks. But sometimes they simply reveal our inability to produce a proof. An independent semantics would allow us to evaluate the truth of assumptions and requirements. SVO was given, indeed based on, such a semantics. We defer discussion of it to Section 6. Another question that arises from this discussion is: just what are the goals of an authentication protocol? We now turn to this question.

4 Authentication Goals

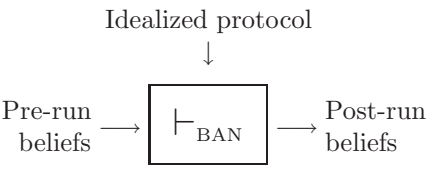
In the previous sections, we have described the syntax of BAN logic [BAN89a] and its descendents, most notably SVO [SvO96], and demonstrated how their axioms and inference rules can be used to derive new information from the factual knowledge specifying a given protocol. For example, this allowed us in Section 2 to construct a formal argument in support of the idea that, by the end of a run of the Needham-Schroeder Shared-Key protocol, the involved parties believe that they share a good key for secure mutual communication. As a matter of fact, this was one of the intended functionalities of this protocol. In general, we will be particularly interested in those derivations that relate the specification of a protocol to its intended goals or requirements.

In the present section, we will be concerned with identifying the authentication goals that a given protocol may be expected to fulfill. Rather than directly attacking this general problem, we will trace the historical development of this quest. We start in Section 4.1 by examining which requirements can be expressed in BAN and then, in Section 4.2, we outline the contributions made by its successors, most notably VO [vO93]. Limitations in these approaches triggered the study of authentication goals per se, independently from their expressibility in any given specification language. Replays, i.e. unwanted behaviors due to the interferences of multiple runs of a protocol, were soon identified as a major cause of unsatisfiable authentication goals, which opened the door to subtle attacks.

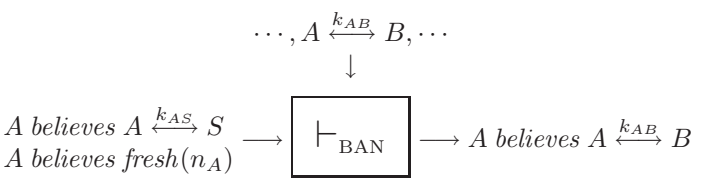
We examine them in Section 4.3. One of the results that emerged from this study is that specification languages such as BAN and SVO are not expressive enough to capture some of the authentication goals aimed at avoiding replays. The study of the notion of authentication continued, sometimes to tragi-comic extremes for most of the 1990's. We conclude in Section 4.4 by listing some of the problems that emerged and the proposed solutions.

4.1 BAN Authentication Goals

We have already outlined the way BAN goes about establishing authentication properties: given assumptions about the beliefs held by a principal before running a given protocol, it allows deducing beliefs that this principal must hold at the end of a run. This formal derivation is guided by the idealized protocol, which enriches the original specification with explicit descriptions of the intended functionality of selected message components, e.g. that a key is fresh or is good for communication between two principals. Although idealization is an essentially manual process and the logical status of the resulting annotations is dubious, the end-product is the vehicle that allows mapping what a principal believes before running a protocol to what he believes afterward, as described by the following diagram.

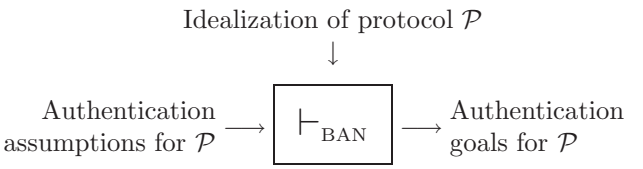


In the case of the Needham-Schroeder Shared-Key protocol examined in Section 2, from assumptions such as that principal A possesses a good key to communicate with a server S ("A *believes* $A \overset{k_{AS}}{\longleftrightarrow} S$" in symbols) and that the nonce n_A is fresh ("A *believes fresh*(n_A)"), we deduced that she can legitimately think that she is handed a good key k_{AB} to communicate with a receiver B (i.e. "A *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$"). The derivation relies on protocol idealizations such as "$A \overset{k_{AB}}{\longleftrightarrow} B$". We can indeed instantiate the above schema in the following partial diagram:



BAN logic does not define the notion of authentication. Instead, it offers means to express the fact that certain properties, clearly related to authentication, should be valid at the end of a message exchange, assuming certain premises hold. We call them *authentication goals*, and use the phrasing *authentication assumptions* for the premises they depend upon. BAN logic views authentication as

a protocol dependent notion: therefore, different protocols will generally require different authentication assumptions and achieve different authentication goals. We schematically describe BAN's approach to authentication in the following diagram, a final evolution of pictures above:



The realization that authentication is a protocol dependent notion leads to our first *observation on authentication*:

1. *There is not a unique definition of authentication that all secure protocols satisfy.*

Although authentication goals depend on the protocol at hands (and on its assumptions), certain goals recur fairly often. In particular, all BAN analyses of key distribution protocols have some of the following formulae as conclusions:

- A *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
- B *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
- A *believes* B *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$
- B *believes* A *believes* $A \overset{k_{AB}}{\longleftrightarrow} B$

Clearly, other goals are possible, e.g. about beliefs concerning public keys. But they do not arise in the examples considered in [BAN89a], in which public keys are always a means to produce session keys in the form of shared keys, rather than the objects about which one would try to establish goals.

On the other hand, there are good key distribution protocols[5] for which some of the above goals are not applicable. Consider for example, the Wide-Mouthed Frog Protocol [BAN89a, CJ97], described below:

Protocol 4 (Wide-Mouthed Frog) [BAN89a, CJ97]

Message 1 $A \to S : \{T_A, B, k_{AB}\}_{k_{AS}}$
Message 2 $S \to B : \{T_S, A, k_{AB}\}_{k_{BS}}$

The initiator A wants to communicate securely with another party B. She achieves this by generating a key k_{AB} and sending it to a trusted server S together with her intention to communicate with B. The server simply forwards this key to B, together with the identity of the generator. The components T_A and T_S are timestamps. The first message is encrypted with a key k_{AS} that

[5] See the end of Section 4.4 below.

A shares with S, which ensures that its contents are not accessible to any other party. Similarly, the second message is encrypted with key k_{BS} that B shares with S. By the end of a run, only A, B and S are expected to know k_{AB}.

Since A generates the key k_{AB}, she has jurisdiction over its freshness and intended use. Therefore

$$A \textit{ believes } A \overset{k_{AB}}{\longleftrightarrow} B$$

is an assumption as well as a goal for the protocol. We leave it to the interested reader to devise the other relevant assumptions of the Wide-Mouthed Frog Protocol 4 as well as the form of its idealization. Provable goals of this protocol include

$$B \textit{ believes } A \overset{k_{AB}}{\longleftrightarrow} B \qquad \text{and} \qquad B \textit{ believes } A \textit{ believes } A \overset{k_{AB}}{\longleftrightarrow} B$$

(again, the formal derivation is left to the reader). It should however be observed that the formula

$$A \textit{ believes } B \textit{ believes } A \overset{k_{AB}}{\longleftrightarrow} B$$

although taken from the above list of typical goals, is not provable. Indeed A cannot hold any belief about B's beliefs since she is not the recipient of any message in this protocol.

4.2 VO Authentication Goals

As reported in Section 2, BAN was shown to have a number of shortcomings, soon to be fixed by a succession of proposals. While early extensions such as GNY [GNY90] and AT [AT91] concentrated on providing a finer modeling language for protocol actions, the logic VO [vO93] also enriched the lexicon of formally stated authentication goals. At the same time, it exposed nuances of the still vague notion of authentication that BAN and its early successors were unable to express. Altogether, VO provided a better understanding of what authentication actually is.

In this section, we present the various forms of authentication available in VO. Rather than trying to be completely faithful to [vO93], we will incorporate some of the adjustments made in later proposals. For simplicity, we will formalize the notions in this section using the syntax of SVO [SvO96], already introduced in Section 3.

We first need to introduce some syntax. VO expresses the fact that a key k is an *unconfirmed secret* for a principal P to communicate with another principal Q as

$$P \overset{k-}{\longleftrightarrow} Q$$

Similarly to BAN's key goodness, this expression means that only P and Q (and third parties trusted by both) know k. It also implies that P has access to this key (e.g. by having received it in a message), but does not enforce a similar requirement on Q: this principal may not be aware of k. It was later observed [SvO96]

that this expression can be given the following simple definition:

$$P \overset{k-}{\longleftrightarrow} Q \quad \equiv \quad (P \overset{k}{\longleftrightarrow} Q \wedge P \; has \; k)^{6}$$

It should be observed that key confirmation is not symmetric. Indeed, $P \overset{k-}{\longleftrightarrow} Q$ is not equivalent to $Q \overset{k-}{\longleftrightarrow} P$.

Given this definition, VO distinguishes the following six forms of authentication:

Ping authentication captures situations where a principal P wants to know whether an interlocutor Q is alive. It is expressed as the following formula, where X can be any message.

$$P \; believes \; Q \; says \; X$$

Observe that not only should Q have uttered something (X), but he should have done so recently, as enforced by the use of *says* as opposed to *said*.

Entity authentication further requires that P's interlocutor Q said something relevant to their present conversation. Given some information Y_P known to be fresh to P (e.g. a nonce), entity authentication mandates that Q recently sent a message $F(X, Y_P)$ from which it is manifest that Q has seen Y_P and has processed it. This is captured by the following formula:

$$P \; believes \; (Q \; says \; F(X, Y_P) \wedge fresh(Y_P))$$

Suitable message transformation functions F must possess the following properties:

- F is effectively one-to-one, by which we mean that for any choice of the arguments X and Y, if $F(X, Y) = Z$, it is computationally infeasible to find values X' and Y' different from X and Y such that $F(X', Y') = Z$. As in [MvOV97], p. 324, the meaning of 'computationally infeasible' is "intentionally left without formal definition", to be interpreted relative to an understood frame of reference. For example it might mean that there is no algorithm that terminates in a time polynomial in the size of the argument of F that computes such X' and Y'. But this is only one possibility.
  This definition also indicates that F genuinely depends on Y_P, in the sense that it is computationally infeasible for an adversary to produce an alteration Y'_P of Y_P that yields the same result, even if he controls the choice of the first argument of F.
- F is computable in practice by Q. This too is left without precise definition. One example would be, given X and Y_P, Q can compute $F(X, Y_P)$ in polynomial time.

[6] In BAN, a principal A could only refer to a key k by believing some property of it, most notably its goodness, or by receiving it in a message. GNY [GNY90] remedied to this deficiency by allowing one to talk about entities possessed by a principal. Here we adopt the AT syntax "*A has k*" introduced in Section 3.

- P can effectively verify that the received value $F(X, Y_P)$ has actually been constructed by using Y_P. This can be achieved in two different ways:
  - P can in practice compute enough of the inverse of F to expose the use of Y_P. This is the case, for example, if $F(X, Y_P) = \{Y_P\}_X$ and X is P's public key.
  - P has access to X (and Y_P), can effectively compute $F(X, Y_P)$, and can verify whether the result corresponds to the value transmitted by Q. An example of this situation is when F is a hash function and X is known to P.

Secure key establishment indicates that a principal P believes that he has a good key k to communicate with a counterpart Q. Given the above notion of unconfirmed secret, this goal is easily expressed by the following formula:

$$P \textit{ believes } P \overset{k-}{\longleftrightarrow} Q$$

Key freshness simply requires that a principal P believes a key k to be fresh:

$$P \textit{ believes } \textit{fresh}(k)$$

Mutual understanding of shared keys applies to situations where a principal P can establish that an interlocutor Q has sent a key k as an unconfirmed secret between the two of them (from Q's point of view). This is formalized by the following formula:

$$P \textit{ believes } Q \textit{ says } (Q \overset{k-}{\longleftrightarrow} P)$$

Key Confirmation is intended to describe scenarios in which a principal P believes that an interlocutor Q has proved to have received and successfully processed a previously unconfirmed secret key k between the two of them. Similarly to the case of entity authentication, we capture the "confirmation" aspect of this definition by requiring Q to return k to P, modulo the application of a function F that is effectively one-to-one, computable in practice by Q, and effectively verifiable by P. We have the following formal definition:

$$P \textit{ believes } (P \overset{k-}{\longleftrightarrow} Q \wedge Q \textit{ says } F(k))$$

It should be observed that key confirmation is not the same as the BAN-style second-order belief "P *believes* Q *believes* $P \overset{k}{\longleftrightarrow} Q$", which may wrongly imply that Q believes that k is a good key for them to communicate. For a similar reason, it differs from mutual understanding of shared keys "P *believes* Q *says* $Q \overset{k-}{\longleftrightarrow} P$".

These definition shed substantial light on the notion of authentication. However, they also raise further questions, a clear indication that VO has moved our understanding of authentication forward, but also that it has not exhausted

the subject. A closer look at entity authentication and mutual understanding of shared keys will reveal some problems, that will be addressed in the rest of this section.

Given actual principals A and B, the intended meaning of the entity authentication goal

$$A \textit{ believes } (B \textit{ says } F(X, Y_A) \wedge \textit{fresh}(Y_A))$$

is that A is engaged in a protocol run with B and she thinks that B said something in response to the nonce Y_A she generated for this run. Observe however that this goal does not impose any constraint on B's assumptions; an intruder could indeed have rerouted messages in such a way that Y_A entered a conversation B was having with a third principal, C say; B may then have freshly sent $F(X, Y_A)$ to C, but the intruder altered the intended course of this message so that it reached A instead. This undesirable behavior passes the above entity authentication test.

Consider now the following goal, an instance of the mutual understanding of shared keys:

$$A \textit{ believes } B \textit{ says } (B \overset{k-}{\longleftrightarrow} A)$$

The concern raised above for entity authentication does not apply here since the presence of the unconfirmed secret expression $B \overset{k-}{\longleftrightarrow} A$ indicates that B is aware of the fact that k is intended to communicate with A. There is however room for attacks: again, an intruder may have rerouted messages so that B thinks that the key k is being used in a run r he is conducting with A, while A believes she is using it in a different run r', although with B. Again, this potentially harmful behavior satisfies the above notion of mutual understanding of shared keys goal.

Both scenarios arise as (intruder-assisted) misunderstandings: the involved principals are participating in an apparently legal run of the protocol, but not in the same run and not necessarily with each other. Both situations involve an interleaving of at least two protocol runs, with the intruder altering the message routes to unduly connect these otherwise independent runs. Such situations are called replays and will be examined in detail in the next section.

4.3 Replay Attacks

As we just discussed, a *replay attack* is characterized by an intruder opportunistically bending the path of messages belonging to different runs of a protocol, possibly after making minor changes to the messages themselves. An in-depth study of the different incarnations of the notion of replay was undertaken in [Syv94]. We present this analysis and use it to measure the expressiveness of the authentication logics from Sections 2 and 3. Two attempts at covering more replay attacks, one that refines the BAN model of time and one that introduces the notion of role, are then discussed.

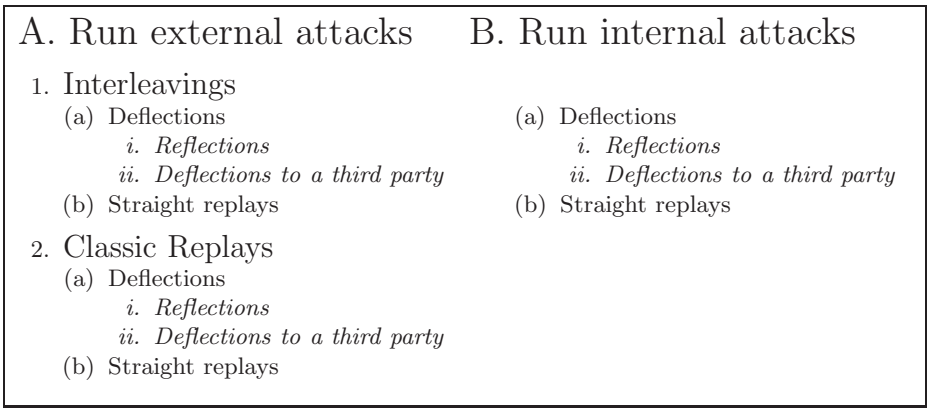


Fig. 2. A full taxonomy of replays [Syv94]

A Taxonomy of Replays

Syverson, in [Syv94], proposes two orthogonal classifications of replays, which formalize the observations that these misbehaviors derive from the interleaving of multiple protocol runs, and that the intruder redirects messages among them, respectively. We will now examine them in detail. The overall combined taxonomy is displayed in Figure 2.

A first way to approach replay attacks is to distinguish them on the basis of which runs the replayed messages are taken from. This materializes in a *run taxonomy* [Syv94], which immediately branches into the following two classes:

A. In a *run external attack*, the replayed message comes from outside the current protocol run. This option involves the execution of at least two runs, which can be either concurrent or sequential, as indicated by the next branching point in this taxonomy:
   (1) An *interleaving attack* requires two or more runs to take place contemporaneously. The intruder uses the different runs in turn as oracles to answer the challenges set forth by the others. A popular example of this form of replays is given by Lowe's attack [Low96] on the Needham-Schroeder Public-Key Authentication Protocol 7, which we will examine in Section 4.4.
   (2) An attack that involves external runs but without the requirement that they should be contemporaneous is called a *classic replay*. The intruder remembers messages sent back and forth during previous runs, and opportunistically replays them to mount an attack on the current run. We have seen an example of classic replay in Section 2 as the Denning-Sacco attack [DS81] on the Needham-Schroeder Shared-Key Authentication Protocol 1.

B. An attack can also result from opportunistically replaying messages from the current protocol run. These are known as *run internal attacks*. An example

involving the Neuman-Stubblebine repeated authentication protocol [NS93, CJ97] has been exposed by Syverson in [Syv93b] and by Carlsen in [Car93].

Another way to look at replay attacks examines which messages are rerouted by the intruder, and how this is done. The resulting classification is known as the *destination taxonomy* [Syv94]. Let us first consider who the replayed message was intended for:

a. The first situation, called *deflection*, redirects the replayed message to a principal different from its intended recipient. This situation can be further refined in the following subcases:
   (i) First, the replayed message can be sent back to its sender. This is called a *reflection attack*.
   (ii) We can also have a *deflection to a third party*, in which the message in question is redirected to a principal that is neither the intended recipient or the originator.

b. An intruder can mount an attack by channeling a message to its intended destination, but with some delay and possibly in a different run of the protocols. This is known as a *straight replay*.

We will now demonstrate the various forms of destination attacks by examining a well-known disruption on the a variant of a draft protocol due to Yahalom, a version of which was ultimately published in [Yah93]. The variant we consider here was first presented in [BAN89a]. By virtue of this iterated genesis, we call it the BAN-Yahalom protocol. It is specified as follows:

Protocol 5 (BAN-Yahalom) [BAN89a]

Message 1	$A \rightarrow B:$	A, n_A
Message 2	$B \rightarrow S:$	$B, n_B, \{A, n_A\}_{k_{BS}}$
Message 3	$S \rightarrow A:$	$n_B, \{B, k_{AB}, n_A\}_{k_{AS}}, \{A, k_{AB}, n_B\}_{k_{BS}}$
Message 4	$A \rightarrow B:$	$\{A, k_{AB}, n_B\}_{k_{BS}}, \{n_B\}_{k_{AB}}$

The initiator A and the responder B rely on a server S to generate a key k_{AB} that would allow them to communicate securely. The long term keys k_{AS} and k_{BS} guarantee the mutual authentication of the server and the principals A and B, respectively. Intentionally, the third message indirectly authenticates B to A by having the server encapsulate both B's identity and A's fresh nonce n_A in the message $\{B, k_{AB}, n_A\}_{k_{AS}}$. The fourth message authenticates A to B by encrypting B's nonce n_B with the newly acquired (and supposedly secure) key k_{AB}.

This protocol is subject to the following attack, first presented in [Syv94], which makes use of three protocol runs, which we distinguish by using different numerals and indentations. The intruder is given the name E (the Eavesdropper), and we write E_P to indicate an action of the attacker while impersonating

principal P. The attack unfolds as follows:

$$
\begin{array}{llll}
1. & & A & \rightarrow E_B: \; A, n_A \\
 & I. & E_B & \rightarrow A \; : \; B, n_A \\
 & II. & A & \rightarrow E_S: \; A, n'_A, \{B, n_A\}_{k_{AS}} \\
 & & ii. \quad E_A & \rightarrow S \; : \; A, n_A, \{B, n_A\}_{k_{AS}} \\
 & & iii. \quad S & \rightarrow E_B: \; n_A, \{A, k_{AB}, n_A\}_{k_{BS}}, \{B, k_{AB}, n_A\}_{k_{AS}} \\
3. & & E_S & \rightarrow A \; : \; n_E, \{B, k_{AB}, n_A\}_{k_{AS}}, \{A, k_{AB}, n_A\}_{k_{BS}} \\
4. & & A & \rightarrow E_B: \; \{A, k_{AB}, n_A\}_{k_{BS}}, \{n_E\}_{k_{AB}}
\end{array}
$$

In line (1), A generates the nonce n_A to communicate with B. The outgoing message is intercepted by E and replayed to A in line (I) after altering its postulated originator to B. In A's view, this is the first message of a different run, with B as its originator, and therefore she responds as expected by generating a nonce n'_A and forwarding the message $(A, n'_A, \{B, n_A\}_{k_{AS}})$ to the server in line (II). The intruder alters this message *en route* by replacing the nonce n'_A with n_A in line (ii). Logically this is part of a third run of the protocol (the server has no reason to suspect that this run lacks its first message). The server performs its task on line (iii) by generating the message $(n_A, \{A, k_{AB}, n_A\}_{k_{BS}}, \{B, k_{AB}, n_A\}_{k_{AS}})$. These two inner runs are left dangling. We return instead to the outer run, where A is expecting a reply from S to her indirect request of line (1) via B. In line (3), the intruder replays the message captured in line (iii) after substituting a nonce n_E of his own in place of the outermost occurrence of n_A. This message has the expected form, and therefore A replies in line (4) as dictated by the text of the protocol.

Although no key is revealed to the intruder E, an attack has taken place since A believes she has been talking to B without this principal even participating in any run. This is clearly a failure of authentication. In order to mount the attack, the intruder makes use of three replay techniques:

- Going from lines (1) to (I), we first have a reflection of the nonce n_A back to A.
- Going from lines (II) to (ii), we have a straight replay of the message components A and $\{B, n_A\}_{k_{AS}}$ across two different runs of the protocol.
- Finally, going from lines (iii) to (3), we have a third party deflection of the encrypted components $\{A, k_{AB}, n_A\}_{k_{BS}}$ and $\{B, k_{AB}, n_A\}_{k_{AS}}$ from S to A and away from B.

Figure 2 integrates the run and destination taxonomies of replays, showing in this way all possibilities for a replay attack. This is therefore a complete classification.

Gauging Expressiveness

The above taxonomy of replays gives a clear view of the different ways an intruder can take advantage of the messages exchanged in one or more runs of a protocol to mount an authentication attack. This minute classification is also

an excellent basis to measure the expressive power of various protocol analysis formalisms: an ideal system would successfully apply to all points in Figure 2. Most proposals cover instead a more spotty spectrum. In this section, we will make use of this taxonomy to outline the strengths and weaknesses of the authentication logics discussed in Sections 2 and 3. The results of this analysis should be taken with a grain of salt: there are cases where a formalism does not have mechanisms to systematically expose a certain class of attacks and yet has tackled specific instances of this class.

We shall first consider BAN logic [BAN89a] introduced in Section 2. Freshness is the only mechanism available in BAN to distinguish a run from another. This is a very weak mechanism indeed, since its effect is limited to temporally partitioning protocol actions into recent (i.e. provably fresh) and old. Therefore, freshness alone cannot hope to reveal run internal attacks, nor any form of interleaving attack. It instead focuses on the portion of the run taxonomy [Syv94] that we have called classic replays.

Although by assumption rather than by analysis, BAN captures a similarly small fragment of the destination taxonomy [Syv94]. First, recall that BAN expects a principal to recognize messages he/she has said. This is equivalent to limiting the scope of the verification process to protocols that are immune to reflection attacks. Second, the notion of a (shared) key k being good for two principals A and B to communicate, '$A \overset{k}{\longleftrightarrow} B$', similarly circumvents deflection-to-third-party attacks. What is left of the destination taxonomy is the category of straight replays and some deflection-to-third-party situations that involve public keys.

In summary, the expressiveness of BAN relative to Figure 2 is limited to the zones marked "straight replays", and the area pertaining to "classic replays" among the run external attacks. In spite of this restricted scope, BAN has been successfully used to perform a large number of analyses.

The logic GNY [GNY90] corrects the inability of BAN to talk about reflection attacks by providing syntax (an asterisk "$*$") to flag a message as "not originated here". The other limitations of BAN remain. Surprisingly they are not addressed by the successors of GNY, namely AT [AT91], VO [vO93], and SVO [SvO96].

We can sum up these observations as follows: none of the discussed logics exhausts or fully expresses the notion of authentication. In particular, since all of them, starting with BAN, are equipped to reason about freshness, we deduce that, in general, authentication problems cannot be reduced to enquiries about freshness. This leads to our second observation on authentication:

2. *Freshness is not rich enough to express all the kinds of authentication.*

Adding Time to Increase Expressiveness

As mentioned above, BAN and its successors rely on a simplicistic view of time that only distinguishes "recent" events from "old" actions. Freshness declarations draws the temporal line separating them, although recent messages almost always pertain to the current run. A finer use of time in protocol analysis was proposed in [Syv93a] with the introduction of the modality $\diamondminus$, read "previously". This allows not only breaking the time-line in more than two segments, but also expressing the fact that event occurrences should have happened according to a certain order. For example, a requirement such as

$$A \textit{ received } \{B, k_{AB}, n_A\}_{k_{AS}} \rightarrow \diamondminus(B \textit{ said } \{A, n_A\}_{k_{BS}})$$

means that if A receives the message on the left-hand side of the implication, then B has previously sent the message on the right-hand side. The added temporal operator has therefore the additional effect of capturing a form of causality between events.

A natural question to ask is whether the addition of the above modality to BAN or SVO is sufficient to address all forms of replay in the taxonomy in Figure 2. The answer is unfortunately negative: at least straight replays are not covered.

In order to demonstrate this point, we will rely on the protocol below, first presented in [Sne91]. The system it models consists of a master computer M and a collection of sensors $S_1, \ldots, S_n$, each controlled by a microprocessor. The master computer periodically queries the sensors. The protocol is aimed at authenticating the order and timeliness of their reports.

Protocol 6 (Snekkenes) [Sne91]

Message 1 $M \rightarrow S_i : \mathsf{Query}(i, j)$
Message 2 $S_i \rightarrow M : \lfloor n_{ij}, \mathsf{Query}(i, j), \mathsf{Answer}(i, j) \rfloor_{k_i^{-1}}$
Message 3 $M \rightarrow S_i : \lfloor n_{ij} \rfloor_{k_M^{-1}}$

In the first message, M sends a query $\mathsf{Query}(i, j)$ to sensor S_i, where j is a progressive number. In the second message, the invoked sensor, S_i returns an answer $\mathsf{Answer}(i, j)$ together with the original query and a nonce n_{ij} aimed at ensuring the freshness of the reply. The origin of this composite message is guaranteed by having S_i sign it with its private key k_i^{-1}. Upon receiving this message, M responds by signing the nonce n_{ij} with his own key k_M^{-1}.

This protocol is not immune to straight replay attacks, even if M keeps track of all used n_{ij} and (correctly) assumes these values are fresh. An intruder can indeed subvert the result of this protocol by intercepting a query $\mathsf{Query}(i, j)$ on its way from M to S_i, forwarding it multiple times to S_i, and letting through to M the most desirable answer.

This attack can be neutralized by reversing the order of the last two messages of this protocol. Consequently, the nonce n_{ij} is now generated by M rather than

by S_i. Moreover, it is now the sensor's duty to memorize the nonces, verify their freshness, and limits its answers to one per nonce, to preclude replays. The master computer shall maintain an association between nonces and queries to prevent the subversive rerouting of signed nonces to sensors different from the one they were intended for.

Snekkenes observed in [Sne91] the rather unsettling fact that the BAN analysis of both variants of this protocol is same. He furthermore proved that a similar limitation holds for any two variants of a given protocol that differ only by the order of the exchanged messages:

Theorem 1. *(Snekkenes '91)*

1. *Let $\mathcal{P}$ and $\mathcal{P}'$ be protocols composed of the same messages, although not necessarily in the same order.*
2. *Assume that $\mathcal{P}$ can be shown to satisfy some goal G given certain assumptions $\mathcal{A}$.*
3. *Furthermore, assume that $\mathcal{P}'$ is demonstrably insecure.*

Then, $\mathcal{P}'$ can also be shown to satisfy the goal G given the assumptions $\mathcal{A}$.

This result was rigorously proved in the context of an annotated sequent calculus for BAN logic.

The above theorem states that extending BAN queries to faithfully account for the causal ordering of protocol actions is not sufficient to prevent all forms of replay attacks. This leads to our third observations on authentication:

3. *Correct causal order and source of a message are not strong enough for all authentications.*

Roles in Cryptographic Protocols

The most visible effect of the introduction of the temporal operator $\diamond$ in the previous section was to extend the language used to express and validate protocol requirement. A similar proposal in [Bie90] focused instead on the language used to specify a protocol.

In [Bie90], protocols are described in the logic of knowledge and time CKT5, which enriches a fragment of first-order logic with the modal operator $K_{A,t}$ and a suitable set of axioms. The intended meaning of a formula of the form $K_{A,t}\ \varphi$ is that at time t principal A knows that φ holds. The use of proper quantifiers over time variables allows capturing the relative temporal ordering of events, similarly to what we have observed with $\diamond$.

The introduction of this modality makes it possible to put strict temporal constraints on the actions that a principal participating in a protocol is allowed to perform. This permits expressing scenarios where, for example, if A sent m_1 and A received m_2, then the next action of A is to send m_3. In this proposal, the protocol actions available to a principal are organized in a *role*, given as

the sequences of message transmissions that this principal is going to perform, possibly in response to the reception of some well-defined messages. A protocol specification is then presented as a set of roles, one for each participating principal. It should be observed that this approach constitutes a radical change of course with respect to the BAN-like specification methodology discussed so far: in these languages, a protocol was described by listing the messages exchanged during an expected run, while roles focus on the individual view of each principal, independently from any run.

The CTK5 specifications given in [Bie90] allowed each honest principal participating in a protocol to play exactly one role. It was shown in [Sne92] that this restriction could give an incorrectly clean bill of health: attacks that relied on having the same principal act both as an initiator and a responder, for example, were missed. This same paper corrected this limitation by upgrading the one-to-one relation between roles and principal proposed in [Bie90] to a many-to-one correspondence. Therefore, a given principal was now associated with a set of roles, an entity also known as a *multi-role*. Differently from roles, multi-roles could, for example, express the necessary conditions to set up the attack on the BAN-Yahalom protocol discussed in Section 4.3.

The CKT5 formalization of roles and multi-roles used in [Sne92] was later simplified in [Car94], which also gave an algorithm to generate CKT5 role specifications from the BAN-like "standard notation" of a protocol.

Clearly, if the protocol at hand is constrained in such a way that every honest principal can play at most one role, then no multi-role flaws can be uncovered. Even in this limited setting, the use of CTK5 as a specification language does not prevent the possibility of all attack. The Snekkenes Protocol 6 from Section 4.3 is subject to the same attack even when expressed in this language. We can therefore strengthen our last observation on authentication as follows:

4. *Correct causal order and message source, and freedom from multi-role flaws are not strong enough for all authentications.*

4.4 A Child's Garden of Authentications

Starting with the most common authentication objectives of BAN logic [BAN89a], the previous section has described the contributions made by various researchers to the formalization and understanding of the notion of authentication. We saw how these original goals were extended in languages such as VO [vO93] and SVO [SvO96]. We then categorized attacks relative to the taxonomy of replays defined in [Syv94] and finally discussed a series of proposal aimed at repairing specification [Bie90, Sne92, Car94] and requirement [Syv93a] deficiencies of the BAN family of logics. Yet, not all attacks could be nailed down.

By this time, we were in the mid 1990s and the notion of authentication was looking like a more and more distant chimera. The research toward this holy

grail intensified, and considerable effort was spent trying to answer the following basic question:

Is there an adequately strong criterion for freedom from replay?

In this section, we will report on some of the progresses that were made toward this elusive goal. We shall anticipate that this question is still open. As we will see in Section 5, this quest is not however as popular as it once was, mainly because several researchers have now given guidelines aimed at constructing protocols that are free from attacks by design.

What Do We Mean by Entity Authentication?

Gollmann raised the question in the title of this section in the homonymous paper [Gol96]. The notion of *entity authentication* had been used liberally, often abused, in the security literature (we gave one of the many definitions in Section 4.2). Gollmann's paper discusses various meanings attributed to this phrase, and crystallizes some of these definitions in the context they ought to be used.

One of the strongest meanings of "entity authentication" requires that all the communications that constitute a session be accessible only to the involved parties, or to some entity in whose integrity they can put a reasonable amount of confidence. This degree of authentication is usually attained by encrypting all the communications between two principals by means of a *session key* freshly generated in a secure manner by a trusted third party. This constitutes the essence of Gollmann's first authentication goal:

G1: The protocol establishes a fresh session key, known only to the session parties and possibly to a trusted server.

While this goal is sufficient when considering protocol runs in isolation, situations that may involve several runs require reinforcing this requirement with the following clause:

G1': Furthermore, compromising old sessions keys does not lead to the compromise of new session keys.

In particular, new session keys should not be transmitted encrypted with old session keys.

A second meaning of "entity authentication" requires that a principal A can ascertain that an interlocutor B has received and successfully interpreted a message sent by A to B. Gollmann expresses this requirement as follows, modulo minor editing:

G2: A key associated with a principal B was used in a message received by another principal A in the protocol run, in a response to a challenge issued by A in the form of a nonce or a timestamp.

This is what we called "entity authentication" in Section 4.2.

A yet weaker form of "entity authentication" simply requires a principal to be able to ascertain that an intended interlocutor was active during a protocol run. This is expressed as the following goal:

G3: A key associated with a principal B was used during the protocol run, in a response to a challenge issued by another principal A in the form of a nonce or a timestamp. However, A did not need to receive a message where this key was used.

This is essentially what we called "ping authentication" in Section 4.2.

Agreements

In [Low97], Lowe observed that all definitions used to talk about authentication have the following form:

> A protocol $\mathcal{P}$ guarantees property X to initiator A for another principal B,
>
> *iff*
>
> whenever A completes a run of the protocol, apparently with responder B, then a certain requirement ψ holds.

We denote the condition "whenever A completes a run of the protocol, apparently with responder B" as φ_{AB}. Then, all the definitions can be seen as implications of the following form:

$$\varphi_{AB} \rightarrow \psi.$$

Here, A and B are parameters rather than specific principals. Therefore, although these goals may appear to be bound to the principals, they are actually more general.

It should be observed that these goals are validated once a run is completed. Therefore, they are intended to authenticate runs, rather than individual messages as in the case of the requirements for BAN examined in Section 4.1.

We will now examine some of the property-requirement pairs (X, ψ) considered in [Low97]. These definitions refine and give a more precise meaning to notions such as ping or entity authentications discussed above.

Aliveness: $\psi =$ "B has been running the protocol".

This requirement extends ping authentication to protocol runs. When satisfied, it guarantees that A's interlocutor, B, has been active some time in the past. Situations in which the run proceeds smoothly from A's point of view without B taking part in any action represent a failure of aliveness. We have observed such a situation in the attack to the BAN-Yahalom Protocol 5 in Section 4.3.

Like every requirement discussed in [Low97], there is a *recent* version of aliveness: $\psi =$ "B has been running the protocol *recently*". Recent aliveness requires B to have been active during the current run. Notice that B does not need to have been running the same protocol as A, and even if he did he may have run it with a different party.

It should be noted that recent aliveness is not only stronger than ping authentication, but it also subsumes VO's entity authentication discussed in Section 4.2. Indeed, recent aliveness is manifested in a run of a cryptographic

protocol by witnessing precisely the transformations required by this form of authentication. From that point of view, recent aliveness possibly gives a meaning to the notion of entity authentication.

Weak agreement: ψ = "B has previously been running the protocol, *apparently with* A".
Weak agreement strengthens aliveness by requiring not only that A's interlocutor B was active, but that A had evidence that he participated in a very direct manner by decrypting or signing messages that he only could have processed (unless the relevant keys were compromised). Observe that weak agreement does not require B to be running the protocol with A, nor can it he assumed to don the expected role (e.g. if A acts as the initiator, B may not necessarily be playing the responder role).

In [Low96, Low97], the difference between (recent) aliveness and weak agreement was illustrated by the attack below, which has achieved world fame and has become a major test bed, sometimes even a rite of passage, for every new protocol verification tool. Lowe's attack operates on the following fragment of a protocol due to Needham and Schroeder [NS78], the public-key version of the Needham-Schroeder Shared-Key Protocol 1 analyzed in Section 2.3.

Protocol 7 (Abridged Needham-Schroeder Public-Key) [NS78]

Message 1	$A \rightarrow B:$	$\{n_A, A\}_{k_B}$
Message 2	$B \rightarrow A:$	$\{n_A, n_B\}_{k_A}$
Message 3	$A \rightarrow B:$	$\{n_B\}_{k_B}$

In this protocol, the initiator A sends her identity and a freshly generated nonce n_A to B, protecting this message by encrypting it with B's public key k_B. Upon receiving it, B generates a nonce of his own, n_B, and sends it to A together with n_A, encrypted with A's public key. In the last message, A sends n_B back to B, encoded with k_B. The protocol originally described in [BAN89a] had an initial key distribution phase in which A and B requested and received the keys k_A and k_B from a trusted server.

Upon completing a run of this protocol, A can be confident that she has been talking with B. Lowe's attack [Low96] shows that this protocol does not provide the reverse assurance. The trace of this attack is as follows:

$$
\begin{array}{lrcl}
1. & A & \rightarrow & E \ \{n_A, A\}_{k_E} \\
\quad i. & E_A & \rightarrow & B \ \{n_A, A\}_{k_B} \\
\quad ii. & B & \rightarrow & E_A \ \{n_B, n_A\}_{k_A} \\
2. & E & \rightarrow & A \ \{n_B, n_A\}_{k_A} \\
3. & A & \rightarrow & E \ \{n_B\}_{k_E} \\
\quad iii. & E_A & \rightarrow & B \ \{n_B\}_{k_B}
\end{array}
$$

On line (1), A starts the protocol with the intruder E, who accesses the contents of the first message, re-encrypts it with B's public key and forwards it to this

principal in line (i). On line (ii), B replies as if the message had come directly from A. The attacker intercepts it and directs it to A in lines (ii) and (2). The initiator A completes the protocol with E by encrypting the nonce n_B she received with E's public key. Finally, E forwards this nonce encrypted with k_B to B. In the end, A (correctly) believes that she has been running the protocol with E, but B is fooled into assuming that he has been talking to A.

It is clear that this attack proves that the Needham-Schroeder Public-Key protocol does not satisfy weak agreement from B's point of view (i.e. after swapping A and B in the above definition). However, it is proved in [Low97] that this protocol satisfies (recent) aliveness.

This attack is routinely used in courses and lectures to support the idea that protocol analysis is difficult, and in seminars and papers to motivate new proposals in this area. It is indeed true that it revealed a novel vulnerability to a protocol published 18 years earlier, and proved correct by a number methods, most notably using the BAN logic [BAN89a]. However, the Needham-Schroeder Public-Key Protocol was never deployed in any real-life setting. More importantly, a careful reading of [NS78] indicates that Lowe's weak agreement for the responder was not among the goals of this protocol.

Among the authors who challenged the legitimacy of this attack, Gollmann [Gol00] observed that it does not reveal any flaw if B's objective in this protocol was to have a communication with A. This corresponds to the notion of ping authentication (Gollmann calls it "authenticating packets"). However, if this protocol was used to establish a secure channel between the two parties, then Lowe's attack is a clear manifestation of a violation. Gollmann called this situation "authenticating circuits".

Non-injective agreement (with respect to data set ds)**:** ψ = "B has previously been running the protocol, apparently with A, *and B was acting as the responder in his run, and both agree on values of variables in ds*".
A yet stronger form of authentication is given by non-injective agreement. Here, A's interlocutor, B, is required to play the expected role, and their runs need to be synchronized to the extent that their respective variables among ds contain the same values. Observe however that this goal does not guaranteed a one-to-one relationship between A's and B's runs (hence the name).
The Needham-Schroeder Public-Key Protocol 7 does clearly not satisfy this requirement. There are however protocols that pass the weak agreement test, but fail non-injective agreement. Examples include the Andrew Secure RPC Handshake [Sat89, CJ97], and Snekkenes Protocol 6 analyzed in Section 4.3.

Agreement: This goal, sometimes called *injective* agreement, reinforces non-injective agreement with the requirement that there is a *one-to-one correspondence between runs*. This last goal in [Low97] forces the runs of each involved party to by fully synchronized, and therefore may appear as the ultimate authentication requirement.
The Wide-Mouthed Frog Protocol 4 presented in Section 4.1 can be proved to satisfy non-injective agreement, but does not pass the stronger agreement

test. An attack that exemplifies this situation is presented in [CJ97]: the adversary replays the server's message to B within the lifetime of the timestamp, essentially acquiring a new timestamp from the server, and repeats this game until A tries to run a legitimate session of the protocol with B, at which point he can replay the appropriate message to B. This attack can be formally expressed in a logic that includes time. A formal analysis in CSP using PVS is given in [ES00].

Intensional Specification

Lowe's hierarchy of authentication goals discussed in the previous section was essentially a response to *intentional specification*, a perhaps overly strict notion of protocol correctness defined by Roscoe in [Ros96]. The definition of intentional specification is as follows:

> A party cannot believe that a run has completed successfully unless a series of messages that agree on all parameters has occurred, up to and including the last message communicated by the given party.

In [Low97], Lowe observes that intensional specification is such a strong requirement that only the most inconsequential behaviors could violated it and yet satisfy agreement. Examples of such failures of intensional specification are:

- Assume that, in response to a request, a server sends a pair of messages (m_A, m_B) to principal A. This party can decrypt m_A, but not m_B, and is expected to forward this component to another principal B, who is able to interpret it. We have seen an instance of this scenario in the BAN-Yahalom Protocol 5 discussed in Section 4.3. Lowe's first example of an intensional specification "attack" that passes the agreement test relies on an adversary that substitutes m_B with some random value X in the message from the server to A. Then, it reinstalls m_B in place of X in the second message from A to B.
- Lowe's second "attack" example takes place in a situation where a server sends messages m_A and m_B to principals A and B, respectively, and in that precise order. An intruder delays the first message so that m_B reaches B before m_A reaches A.

It has been debated whether these failures can reasonably be seen as attacks, in any even remotely practical meaning of the term. In particular, it is not clear whether there are "real" attacks that satisfy agreement but not intensional specification. These doubts are highlighted by analyzing the following previously unpublished protocol.

Protocol 8 (Unpublished)

Message 1	$A \to B:$	$\{n_A, A, B\}_{k_{AB}}$
Message 2	$B \to A:$	$\{n_B, n_A, A, B\}_{k_{AB}}$
Message 3	$A \to C:$	$\{A, C, (n'_A \oplus n_B)\}_{k_{AC}}$
Message 4	$C \to A:$	$\{n_C, A, C, (n'_A \oplus n_B)\}_{k_{AC}}$
Message 5	$A \to B:$	$\{n_B, (n''_A \oplus n_C), A, B\}_{k_{AB}}$

Principals A and B set up a mutual challenge involving nonces n_A and n_B in line (1) and (2). In line (3) and (4), a similar process occurs between A and C, but the fresh value in the third message is not properly a nonce, but the result of taking the X-OR of B's nonce n_B from line (2) and some newly generated nonce n'_A. In the last message, A answers B's challenge from line (2), but also includes one of these pseudo-nonces, which is obtained by taking the X-OR of C's nonce n_C and yet another nonce n''_A generated by A.

Although we did not conduct a formal proof, this protocol seems to satisfy Lowe's notion of agreement. There are however situations in which it violates Roscoe's intentional specification:

- Suppose that A sends the message in line (3) before receiving the nonce n_B in line (2): she could for example use $n'_A = n^*_A$ to form this message without taking any X-OR. While the one-to-one correspondence between runs is not affected, intensional specification is violated since C would receive the nonce n'_A rather than the pseudo-nonce $(n'_A \oplus n_B)$. This may be potentially harmful since the causal relation of messages appears to be affected.
- Suppose now that A generates a nonce n^*_A before receiving B's nonce in line (2), waits for n_B, and only then calculates $n'_A = n^*_A \oplus n_B$ and sends the message in line (3). Now intensional specification seems to be satisfied. However, the end-result is identical since, being X-OR associative and idempotent, $(n^*_A \oplus n_B) \oplus n_B = n^*_A$. Indeed, the value sent to C has been decided before receiving B's message.
- Last, consider an identical scenario, but in which A generates n^*_A *after* receiving B's message in line (2), but without using n_B. Now, the causal relation between the messages is clearly respected, yet C will receive a value that is independent from the nonce n_B.

Similar "attacks" can be constructed with respect the the messages on lines (4) and (5).

Matching Histories

Matching histories [DvOW92] is an older proposal whose strength fits between Lowe's (injective) agreement and weak agreement. This characterization of authentication is particularly interesting because its definition was developed by industrial specialists in secure system design and cryptography rather than by formal methods experts, as for the proposals discussed so far. In particular,

their focus was likely to be on a more practical articulation of the notion of "authentication" geared toward actual applications rather than on mapping out the theoretical terrain.

A protocol satisfies matching histories if the following condition can be proved to hold:

> When a principal A accepts the other party's identity (before receiving or sending further messages), the other party's record of the partial or full run matches A's (with the same values for all message variables).

This requirement is as strong as Lowe's (injective) agreement insofar as the number of runs and all variables must match between A and B. It is however not as powerful as weak agreement since B does not need to have been running the protocol with A. It can however be shown that matching histories and agreement are equivalent if every message exchanged in a protocol includes the identities of the apparent sender and of the intended recipient.

It is interesting to observe that matching histories is motivationally similar to VO's key confirmation (see Section 4.2),

$$P \textit{ believes } (P \overset{k-}{\longleftrightarrow} Q \wedge Q \textit{ says } F(k))$$

while Lowe's various "agreements" goals are motivationally similar to VO's mutual understanding of shared keys (see Section 4.2) and to BAN's second-order belief (see Section 4.1),

$$P \textit{ believes } Q \textit{ says } (Q \overset{k-}{\longleftrightarrow} P) \qquad\qquad P \textit{ believes } Q \textit{ believes } (Q \overset{k}{\longleftrightarrow} P).$$

Cautionary Note

By the end of the 1990s, the research on issues related to authentication had proliferated to the point that some practitioners started noticing a dichotomy between the problems addressed in the academic literature on security, and the solutions sought in real world scenarios. Gollmann, again, voiced these concerns in the paper [Gol00]. He observed that the research in this area was often fueled by a perceived informality in protocol analysis, and, putting it in his own words,

> *[this] motivates the presentation of a new formalism for the analysis of authentication protocols, and the biggest prize to be won is the detection of an attack hitherto unreported. We will argue that such exercises in formal analysis more often add to the problem than help in its resolution.*

Furthermore:

> *Perceived problems with authentication are caused by intuitive but imprecise interpretations of the objective of "authentication", and by neglecting to take into account the environment a protocol is intended to operate in. In many cases, new attacks do not expose subtle flaws in protocols but differences in assumption about protocol goals.*

However, sometimes they do expose subtle flaws. Furthermore, new theories sometimes do turn out to be practically useful. Clearly, this is not always the case, but even then, they often have an impact on our understanding of the various concepts that contribute to what we call security. In these cases (and many other), it is essential not to mistake theoretical results for applied ones, or vice versa.

5 Design Principles and Protocol Logics

At this point we take a brief holiday from formal characterizations of authentication to consider protocols from a more informal and more applied perspective.

5.1 Protocol Design Principles

Abadi and Needham set out "prudent engineering practices for cryptographic protocols" in [AN94, AN96]. These are rules of thumb for good protocol design. They are not meant to apply to every protocol in every instance, but they do provide a laundry list of things that should be considered when designing a protocol. The paper contains useful examples and discussion of the principles. We quote from [AN96] just the principles here and then briefly comment on them below.

Principle 1. Every message should say what it means: The interpretation of the message should depend only on its content. It should be possible to write down a straightforward English sentence describing the content—though if there is a suitable formalism available, that is good too.

Principle 2. The conditions for a message to be acted upon should be clearly set out so that someone reviewing the design may see whether they are acceptable or not.

Principle 3. If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

Principle 4. Be clear as to why encryption is being done. Encryption is not wholly cheap, and not asking precisely why it is being done can lead to redundancy. Encryption is not synonymous with security, and its improper use can lead to errors.

Principle 5. When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message. On the other hand, it is proper to infer that the principal that signs a message and then encrypts it for privacy knows the content of the message.

Principle 6. Be clear about what properties you are assuming about nonces. What may do for ensuring temporal succession may not do for ensuring association—and perhaps association is best established by other means.

Principle 7. The use of a predictable quantity (such as the value of a counter) can serve in guaranteeing newness, through a challenge-response exchange. But if a predictable quantity is to be effective, it should be protected so that an intruder cannot simulate a challenge and later replay a response.

Principle 8. If timestamps are used as freshness guarantees by reference to absolute time, then the difference between local clocks at various machines must be much less than the allowable age of a message deemed to be valid. Furthermore, the time maintenance mechanism everywhere becomes part of the trusted computing base.

Principle 9. A key may have been used recently, for example to encrypt a nonce, yet be quite old, and possibly compromised. Recent use does not make the key look any better than it would otherwise.

Principle 10. If an encoding is used to present the meaning of a message, then it should be possible to tell which encoding is being used. In the common case where the encoding is protocol dependent, it should be possible to deduce that the message belongs to this protocol, and in fact to a particular run of the protocol, and to know its number in the protocol.

Principle 11. The protocol designer should know which trust relations his protocol depends on, and why the dependence is necessary. The reasons for particular trust relations being acceptable should be explicit though they will be founded on judgment and policy rather than on logic.

5.2 Design Principle Comments

Such rules of thumb should always be considered when designing a protocol and only violated when the violation is consciously done for a superseding reason. Since the rules are generally quite compelling, we focus on some of the ways in which they might not apply, as a caution against applying them blindly. (Comments in this section are mostly drawn from [Syv96].)

Building on the above principles, Anderson and Needham set out further principles specifically focused on public-key protocols. Their first principle is an expansion of Principle 5 above.

> Sign before encrypting. If a signature is affixed to encrypted data, then one cannot assume that the signer has any knowledge of the data. A third party certainly cannot assume that the signature is authentic, so non-repudiation[7] is lost. ([AN95], p. 237, Principle 1)

This is a nice principle for illustrating limitations: there are many places where non-repudiation may not be of paramount concern; thus the principle may be too narrow. For example, anonymity may take priority over non-repudiation. This would occur in voting protocols, and in digital cash. Digital cash often makes use of a blind signature, in which the authority issuing the cash signs a 'coin' that has been 'blinded' so that the authority cannot recognize the specific coin and thus tie it to the principal to whom it was issued. After signing the blinded coin, the principal unblinds it so that anyone can recognize it as a coin authentically signed by the issuer.[8]

[7] The goal of non-repudiation is to prevent a principal from denying some action s/he has taken, such as sending or receiving a message.

[8] This is a very simple description. Blinding was invented by Chaum [Cha83]. More on digital cash and other applications of blinding can be found in [Sch96].

This principle may also be too broad: signing encrypted data may be necessary for non-repudiation. One place this can be seen is in a coin-flip protocol. A principal signs encryptions of "Heads" and "Tails" and later reveals the encryption key. Part of the reason is so that she cannot deny the choices offered and also so that the opposing principal cannot deny the choice made. A simple coin-flip protocol protocol demonstrating this point was given in [Syv96]. What follows is an even more simple version of this (without, e.g., replay protection). Other similar protocols were discussed in [Tou92].

Protocol 9 (Simple Coin Flip)

Message 1	$A \rightarrow B:$	$\lfloor \{Heads\}_k, \{Tails\}_k \rfloor_{k_A^{-1}}$
Message 2	$B \rightarrow A:$	$\lfloor X \rfloor_{k_B^{-1}}$ (where X is one of $\{Heads\}_k$ or $\{Tails\}_k$)
Message 3	$A \rightarrow B:$	$\lfloor k \rfloor_{k_A^{-1}}$
Message 4	$B \rightarrow A:$	$\lfloor k \rfloor_{k_B^{-1}}$

Non-repudiation is a fairly subtle requirement. It may be unsurprising that principles such as the one under discussion are subject to the cautionary remarks we have been making. Explicitness, however, would seem to be paramount in all security protocols, and especially in authentication protocols. Indeed, Abadi and Needham regard it (as embodied in PRINCIPLES 1 and 2 above) as the overarching principle in the design of secure cryptographic protocols. It is therefore surprising that there are authenticated key distribution protocols that can only function in the absence of explicitness (especially explicitness as in PRINCIPLE 10). We now present such a protocol.

Protocol 10 (EKE — Encrypted Key Exchange) [BM92, BM93]

Message 1	$A \rightarrow B:$	$A, \{k_A\}_P$
Message 2	$B \rightarrow A:$	$\{\{k_{AB}\}_{k_A}\}_P$
Message 3	$A \rightarrow B:$	$\{n_A\}_{k_{AB}}$
Message 4	$B \rightarrow A:$	$\{n_A, n_B\}_{k_{AB}}$
Message 5	$A \rightarrow B:$	$\{n_B\}_{k_{AB}}$

The idea of the EKE protocol is to function as a privacy multiplier. Let Alice be some client and Bob a server for which Alice has password P. P is thus a secret shared between A and B, and the only means of authentication A possesses. She encrypts a public key k_A with P and sends it to Bob. Bob generates a session key k_{AB} and encrypts this with k_A and then encrypts the result with P. There is then a handshake that shows fresh possession of the session key. The important thing to observe about this protocol is that the content of messages cannot be confirmed upon receipt since the recipient of a message cannot tie its content to any known values until s/he completes the protocol. In particular, principals

cannot tell if received messages have the correct form for them to take the next step. It is only when a recipient gets his last message that he can confirm that the preceding messages had the correct content and acting upon them was appropriate. If any of the messages contained adequate redundancy in content or coding for a principal to know what s/he is receiving (or sending) before the end of the protocol, then the protocol would be vulnerable to guessing attacks since P is a weak secret.

Even if this protocol is a counterexample to the complete generality of explicitness, it is also an example for another of the design principles; Anderson and Needham warn

> Be careful when signing or decrypting data that you never let yourself be used as an oracle by your opponent. ([AN95], p. 240, Principle 3)

EKE puts a spin on that principle; instead of preventing use of principals as oracles it ensures that the output of such oracles is of no use to the attacker.

Despite such unusual examples, explicitness is very often exactly what is required. We now delve deeper into its implications.

5.3 Fail-Stop Protocols

For any definition of authentication, almost all of the failures in the literature are due to active attacks in which a message is somehow altered or substituted for another in a way it was not intended. Thus, stopping such attacks would go a long way towards a general guarantee of protocol security. Fail-stop protocols [GS98] are designed to meet this goal.

Using Lamport's definition of causality [Lam78], we can organize the messages of a protocol into an acyclic directed graph where each arc represents a message and each directed path represents a sequence of messages. In a fail-stop protocol, if a message actually sent is in any way inconsistent with the protocol specification, then all those messages that come after this altered message on some path in the graph (i.e., they are causally after the altered message) will not be sent. Obviously conditions to act upon all protocol messages must be explicit in the content and format of each message in order for the protocol to be fail-stop.

A protocol is said to be *fail-stop* if any attack interfering with a message in one step will cause all causally-after message in the next step or later not to be sent [GS98].

No definition of authentication given so far is sufficient for fail-stop. The main reason is that the definitions we have discussed are focussed on properties that must hold if and when a principal has completed a protocol run. But, fail-stop is a requirement that must hold as the protocol executes. For example, consider the EKE protocol of the last section. This is quintessentially not fail-stop. A principal cannot confirm anything about the content or possibly even encoding of any message until s/he has received the last message of the protocol run.

Claim 1 *Active attacks cannot cause the release of secrets within the run of a fail-stop protocol.*

Claim 1 follows immediately from the definition of a fail-stop protocol, because active attacks do not cause more (or different) messages to be sent; so an attacker using active attacks cannot obtain more secrets than one using passive eavesdropping.

One of the desirable features of a fail-stop protocol is this form of immunity to active attacks. More generally, since an active attack will cause a fail-stop protocol to halt, in a fail-stop protocol no principal will ever produce encryptions or any other computations on data from a message that was not entirely legitimate. Therefore, we need to consider only passive attacks in which an adversary records messages and tries to compute secrets from them. Such passive attacks (and protection measures against them) are much better understood than active attacks and easier to analyze. And, as already noted, they are substantially less common in the attack literature.

This shows us the beginnings of a synergy between design principles and formal analysis, except that fail-stop is not quite a design principle. But the synergy can be strengthened via explicitness based on the principles of Abadi and Needham.

One of the ways to make a protocol fail-stop is to design it in accordance with the following criteria:

1. The content of each message has a header containing the identity of its sender, the identity of its intended recipient, the protocol identifier and its version number, a message sequence number, and a freshness identifier.
2. Each message is encrypted under a key shared between its sender and intended recipient.
3. An honest principal follows the protocol and ignores all unexpected messages.
4. A principal halts any protocol run in which an expected message does not arrive within a specified timeout period.

Here a freshness identifier can be a timestamp (if clocks are assumed to be securely and reliably synchronized) or a nonce issued by the intended recipient. But, the freshness identifier in the first message of the protocol cannot be a nonce since the recipient must be able to determine if the protocol should proceed based on it. So, it must be a sequence number, timestamp, or something that will meaningfully indicate freshness to the recipient. When a freshness identifier takes on a more complicated form, the rules for reasoning about freshness in sections 2 and 3 can be used to determine if the identifier is fresh with regard to the recipient. Basically, these rules say that, if x is deemed fresh and y cannot be computed (in a computationally feasible way) by someone without the knowledge of x, then y is also deemed fresh. Encryption with a shared key in item 2 of this claim can be replaced by the use of an encryption using the recipient's public key of a signature using the sender's private key. We can offer no formal proof of the claim, but it should be clear by inspection.

It might seem that fail-stop protocols automatically guarantee authentication.

Protocol 11 (Simple Fail-Stop Example)

Message 1	$A \rightarrow B:$	$\{A, B,$ *Prot_name, version, seq.*$= 1, T_A,$ *Query*$\}_{k_{AB}}$
Message 2	$B \rightarrow A:$	*Response.*

In the first message T_A is a timestamp, and other fields have their obvious meaning. This message clearly follows the format of above design criteria. The second message is not of that format, but since it is the last one in the protocol, there are no causally-after messages. Thus, the protocol is fail-stop. However, the second message is not authenticated (according to virtually any definition).

Extensible Fail-Stop Protocols

A protocol can be fail-stop even if it contains messages that could have come from any principal at any time. In this section we explore a strengthening of the fail-stop concept.

A message in a protocol is *last* if no protocol message is causally after it. A protocol is *extensible fail-stop* (EFS) if adding any last message to the protocol results in a fail-stop protocol.

Note that limiting to "ping-pong" protocols (where each message is followed by a single successor) implies a unique last message. This is the typical case for two party authentication protocols. The example of Protocol 11 is not EFS because adding a another message after Message 2 would result in a protocol that is not fail-stop. For EFS protocols, authentication is in fact automatically guaranteed—but only message authentication. An example of an EFS protocol is as follows:

Protocol 12 (Simple EFS Example)

Message 1	$A \rightarrow S:$	$\{A,$ *S, Prot., vers., seq.*$= 1,\ T_1,$ *request*$(A, B)\}_{k_{AS}}$
Message 2	$S \rightarrow A:$	$\{S,$ *A, Prot., vers., seq.*$= 2,\ T_2, (k, A, B)\}_{k_{AS}}$
Message 3	$A \rightarrow S:$	$\{A,$ *S, Prot., vers., seq.*$= 3,\ T_3, (k, A, B)\}_{k_{AS}}$
Message 4	$S \rightarrow B:$	$\{S,$ *B, Prot., vers., seq.*$= 4,\ T_4, (k, B)\}_{k_{BS}}$
Message 5	$B \rightarrow S:$	$\{B,$ *S, Prot., vers., seq.*$= 5,\ T_5, (k, B)\}_{k_{BS}}$

This example demonstrates that fail-stop, even extensible-fail-stop, does not imply that the protocol satisfies all kinds of authentication. In the example, all messages are authenticated, but Bob does not know with whom he shares a key. Even a protocol in which Bob is given the wrong name for the principal meant to share the key could still be EFS.

Roughly, most of the authentication properties discussed in Section 4 are properties established by a complete protocol run about messages and the content of messages sent during that run. But, (extensible) fail-stop properties are

authentication properties established by messages about the complete protocol run. For example, the following claim is immediate.

Claim 2 *Extensible fail-stop protocols are immune to replay.*

There is another, potentially more interesting property of EFS protocols.

Claim 3 *The sequential and parallel composition of EFS protocols is extensible fail-stop.*

The claim is justified by cases. For parallel composition: a message inserted causally before a last message of a fail-stop protocol will be ignored or cause a halt. Thus, it will not cause an EFS protocol to cease to be EFS. For sequential composition: let Pr_1 and Pr_2 be two EFS protocols. Suppose that some or all of the messages of Pr_1 are received after a last message of PR_2. If the first message of Pr_1 causes the result to be non-EFS, then Pr_2 was not EFS. (Contradiction.) And, if any later message causes the result to be non-EFS, then Pr_1 was not EFS.

We have already seen that fail-stop protocols need only to be examined for secrecy in the context of passive attacks (because active attacks cannot cause the release of secrets). In addition to its inherent interest, Claim 3 provides another design advantage of EFS protocols. Even analyses that consider interleaving typically assume only one protocol is running. If protocols are EFS, we are free to run multiple protocols in one environment without concern for interleaving attacks.[9]

As noted above, EFS protocols can be simply designed using basic explicitness rules. EFS protocols more flexible wrt composability, and EFS rules simplify the analysis task by removing replay considerations. How else might design rules synergize with protocol analysis logics?

5.4 Design Rules and Protocol Logics

A straightforward way for design rules to synergize with protocol logics is to build design checks directly into the logic. Brackin did precisely that in [Bra00]: he designed the logic BGNY [Bra96], based on GNY. He later developed an associated automated HOL tool, AAPA (Automated Authentication Protocol Analyzer) [Bra98], and a specification language similar to Millen's CAPSL [DM00, Mil]. The resulting system appears to be easy to use. Brackin has analyzed the entire Clark-Jacob library[10] using AAPA. He has also analyzed large commercial protocols such as the Cybercash main sequence protocol [Bra97]. This alone makes his a significant body of work, although we are not primarily concerned with automated tools in this paper.

[9] See Section 7 for a cautionary note.

[10] The Clark-Jacob library is a fairly comprehensive list of known attacks on published authentication protocols [CJ97].

Wedel and Kessler devised another BAN logic we will call 'WK' [WK96]. WK works with an automated tool AUTLOG based on Prolog. One advantages of the WK approach is that no formulae occur in messages. This is another step in solving the problem of the informal nature of idealization. Of course there is still the need for interpretation assumptions (as they were called in Section 3.4). That part cannot be automated. However, analysis in WK automates derivation of the comprehension assumptions. Recall that these were the assumptions that allowed us to express what a principal understands of received messages even though some of the message may be unfamiliar or not decryptable by the principal. In fact, the WK notation for not-understood messages motivated the notation given above in Section 3, although the use by Wedel and Kessler is not exactly the same. Another automated tool in the BAN family is the recent C3PO of Anthony Dekker [Dek00]. This is a GUI tool based on the Isabelle theorem-prover. The logic associated with this tool is called 'SVD', and, like WK, it is a variant on SVO. Neither of these has the published track record of analyses of Brackin's work, however.

A different approach to automation that again combines logics and design is that taken by Clark and Jacob in [CJ00]. In some sense the idea of this approach is to not do design at all. Rather goals are stated and then protocols are synthesized that meet these goals. Clark and Jacob automatically generate protocols from basic BAN logic goals (as described in Section 4.1) using genetic algorithms and simulated annealing. Another automated synthesis, but based on Song's Athena model checker rather than on BAN, was presented by Perrig and Song in [PS00]. Related ideas can be found in [Gut]. This no-design approach may have great long term potential, but it is still early. As we have seen, even simple protocols are subtle and the contribution of such approaches may be to produce protocols with desirable features that no person would be likely to design.

Buttyán, Staamann, and Wilhelm also synthesize protocols from a BAN-like logic [BSW98]. However, unlike the previously mentioned approaches that effectively generate random protocols and then prune to the results that meet desired goals, they directly synthesize protocol designs from goals. Their protocol designs are slightly more abstract than we have been considering. They specify and reason about protocols on the more abstract level of channels. The encryption mechanism used to secure the channel is regarded as an implementation issue. The result is thus somewhat similar to spi calculus [AG99] but is closer to Needham-Schroeder style specifications. Roughly speaking, their design logic synthesis rules work by running an abstracted version of BAN in reverse. For example, if C is a channel, then the following is a synthesis rule.

$$\frac{P \textit{ believes } P \textit{ received } X \textit{ on } C}{P \textit{ sees } X \textit{ on } C \qquad P \textit{ can read } C}$$

A protocol that would satisfy the goal above the line would need to have P receiving X on channel C, where C might be, e.g., encryption using P's public key or a key P shares with another principal.

These rules give articulated goals, not conclusions. In some cases they yield intermediate goals that require further applications of rules before the protocol can be synthesized.

A common theme of this design logic and the synthesis tools is that one first specifies what is wanted then looks at the protocol. That means that one must state generic requirements in a formal language. The intuitive expression of requirements is thus a strong advantage of the logical approach. This also suggests another way of combining formal requirements statements with existing formal analysis techniques: Give the semantics of requirements language in the language of the formal analysis method. Then, use the formal analysis method to evaluate the truth of requirements statements in models of the protocol.

6 Semantic Approaches

We have seen in the previous section that it is often advantageous to use distinct languages to express a protocol under investigation and the goals it is expected to meet. The protocol specification language typically has an operational flavor that makes it particularly adequate for analyses based on simulation, such as model checking. Requirements are more easily stated in declarative formalisms, preferably with strong logical foundations. In order to be usable, requirements need to be mapped down to the execution model supported by the protocol specification language. We do so by endowing the requirement logic with an operational semantics in terms of the formalism used to express the protocol.

In this section, we will briefly examine two instances of this symbiosis. First, in Section 6.1, we look at the successful NRL Protocol Analyzer [Mea94, Mea96] together with the NPATRL requirements logic [SM96]. Then, in Section 6.2, we discuss a recently proposed synergy that adopts the popular strand formalism [THG97, THG98b] as an operational model and a BAN-like logic as the specification formalism [Syv00].

6.1 NPATRL

Our first case study will consist of the established synergy between the NRL Protocol Analyzer [Mea94, Mea96] and the NPATRL requirement language [SM96]. We first sketch relevant aspects of the NRL Protocol Analyzer and then introduce NPATRL.

The NRL Protocol Analyzer Model

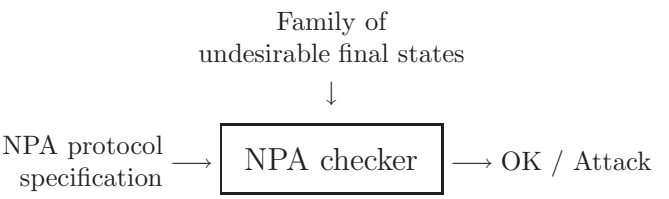
The NRL Protocol Analyzer, or NPA for short, is a computer-assisted verification tool for security protocols which combines model checking and theorem-proving techniques to establish authentication and secrecy properties. We will limit the presentation of this system to the aspects that will be relevant to our discussion of the NPATRL language. The interested reader is invited to consult [Mea94, Mea96] for further details.

A protocol is modeled as a number of communicating state machines, each associated with a different roles. Their transitions correspond to the actions that comprise the corresponding role. At run time, roles are executed by *honest principals* who faithfully follow the protocol. Several instances can be executing at the same time, and they are distinguished by means of a unique round number.

The intruder is modeled after the Dolev-Yao adversary, described in Section 1.1. *Dishonest principals* share their keys and other confidential information with the adversary.

The messages in transit, the information held by each principal and the intruder, the runs currently being executed, and the point that each of them has reached constitute the global *state* of the NRL Protocol Analyzer. A protocol action implements a local transformation with global effects on the state. The initial state is implicit in the protocol specification.

In order to verify a protocol, a specification is fed into the run-time system of the NRL Protocol Analyzer together with the description of a family of states that correspond to attack situations. The system applies protocol actions backwards from these target states until it either reaches the initial state, or it exhausts all possibilities for doing so. In the first case, it reports the sequence of transitions that link these two states: this tracks a possible attack. The second case establishes that an attacker cannot produce the target scenario. Although the search space is in general infinite, the NRL Protocol Analyzer incorporates techniques based on theorem proving that have the effect of soundly restricting the search to a finite abstraction, in most cases. We can pictorially describe the operations of the NRL Protocol Analyzer by means of the following diagram, where we have kept the fairly stable intruder model implicit:



As it regresses back towards the initial state, the NRL Protocol Analyzer maintains a *trace* of the sequence of actions that, when executed, lead to the target state. If the initial state is ever reached, the sequence constructed in this manner is returned as a description of the attack it has found. When a path is abandoned, the corresponding trace fragment is discarded. Traces are sequences of *events* of the following form:

$$event(P, Q, T, L, N)$$

In general, any protocol or intruder state transition may be assigned an event. The arguments are interpreted as follows: P is the principal executing the transition, Q is the set of the other parties involved in it, T is a name that identifies the transition, L is a set of relevant words, and N is the local round number of

the transition. There are three categories of events which correspond to receiving a message (predicate "receive"), accepting data as valid as a result of performing certain checks (predicate "accept"), and sending a message (predicate "send"). Here are two examples:

$$\text{accept}(\text{user}(A, \text{honest}), [\text{user}(B, H)], \text{initiator\_accept\_key}, [K], N)$$
$$\text{send}(\text{server}, [\text{user}(A, \text{honest}), \text{user}(B, \text{honest})], \text{server\_send\_key}, [K], N)$$

The first event describes the execution of a transition called "initiator_accept_key" by honest principal A that involves a key K and some other principal B who may or may not be honest. The second event records a server's application of rule "server_send_key" relative to honest principals A and B, and key K.

Any principal can perform a "send" or a "receive" event, but only the honest principals are entitled to do an "accept" event. As we will see below, events are the building blocks of the NPATRL language.

A Requirement Language for the NRL Protocol Analyzer

The NRL Protocol Analyzer model described above has successfully been used to verify a number of protocols, sometimes uncovering previously unknown flaws [Mea94, Mea96]. This is all the more laudable once we acknowledge the implicit and rudimentary manner in which requirements are entered in this system: secrecy and authentication goals are expressed as states that should not be reachable from the initial state. This unintuitive and occasionally error prone way of writing requirements would have made it very difficult to use the NRL Protocol Analyzer for large protocols.

The *NRL Protocol Analyzer Temporal Requirements Language*, better known as NPATRL (and pronounced "N Patrol"), was designed to address these shortcomings [SM96]. This formalism makes available the abstract expressiveness of a logical language to specify requirements at a high enough level to capture intuitive goals precisely, and yet it can be interpreted in the NRL Protocol Analyzer search engine.

NPATRL requirements are logical expressions whose atomic formulas are *event statements*: they include the "receive", "accept" and "send" events that can be found in the trace of an NRL Protocol Analyzer search, and the special "learn" event that indicates the acquisition of information by the adversary. The logical infrastructure of NPATRL consists of the usual connectives $\neg$, $\wedge$, $\rightarrow$, etc, and the temporal modality $\diamondsuit$ which, similarly to what we saw in Section 4.3, is interpreted as "happens before" or "previously".

For example, we may have the following requirement:

> *If an honest principal A accepts a key K for communicating with another honest principal B, then a server must have previously generated and sent this key with the idea that it should be used for communications between A and B, and that both are expected to be honest.*

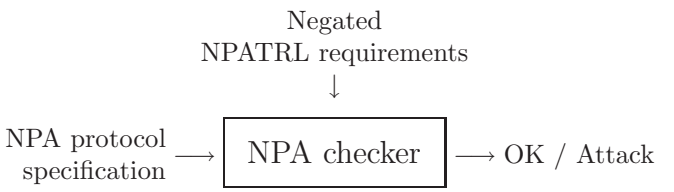
We can use the NRL Protocol Analyzer events given in the previous section to construct an NPATRL formula that expresses it:

$$\begin{aligned} &\text{accept}(\text{user}(A, \text{honest}), [\text{user}(B, H)], \text{initiator\_accept\_key}, [K], N) \\ \rightarrow\ &\Diamond\!\!-\ \text{send}(\text{server}, [\text{user}(A, \text{honest}), \text{user}(B, \text{honest})], \text{server\_send\_key}, [K], N) \end{aligned}$$

This formula is a simple expression of the above requirement. A direct encoding in terms of final states is tricky, in particular if we want to faithfully express the temporal meaning of the operator "$\Diamond\!\!-$".

Intuitively, the protocol verification process changes from what we discussed in the previous section by using NPATRL requirements where the final state appeared. More precisely, we first need to map every NPATRL event statement to an actual event in the NRL Protocol Analyzer specification of the protocol. Then, we take the negation of each NPATRL requirement as a way to characterize the states that should be unreachable if and only if that requirement is satisfied. At this point, we perform the analysis as in the previous section: if the NRL Protocol Analyzer proves that this goal is unreachable, the protocol satisfies the original requirement. Otherwise, it returns a trace corresponding to a attack on the protocol that potentially invalidates the requirement.

Abstractly, the verification process of the NRL Protocol Analyzer enhanced with the NPATRL language can be expressed by the following diagram:



NPATRL has been extensively used in the last few years to analyze protocols with various characteristics. Among these, generic requirements have been given for two-party key distribution protocols [SM93, SM94] and two-party key agreement protocols [SM96]. The most ambitious specification undertaken using NPATRL has involved the requirements of the credit card payment transaction protocol SET (Secure Electronic Transactions) [MS98]. SET proved particularly difficult to specify for several reasons. First, nowhere in its hefty documentation (indeed, about 50cm thick) [SET97] are the requirements of this protocol stated, even informally. Second, it relies on some unfamiliar constructs such as dual signatures. Finally, the objects to be authenticated are dynamic: unlike keys, what is agreed upon changes as it passes from one principal to another. This exercise revealed several ambiguities [MS98].

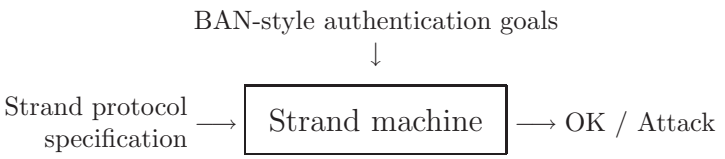
6.2 Strand Semantics for BAN Languages

In the last section, we presented a case study that separated the syntax in which requirements are best stated (NPATRL) from the semantics in which the

protocol is best specified and evaluated (NPA). In this section we explore the possibility of a similar strategy for BAN-style languages. (The content of this section is largely taken from [Syv00].)

Some BAN-like logics already have a model-theoretic semantics, for example, AT and SVO. Such a semantics can provide assurance in the reasoning embodied in a logic, via a soundness result. However, as illustrated in the last section, it can also provide another level on which to reason. These points were alluded to in Sections 2.4 and 3.5. And, providing an independently motivated model-theoretic semantics for BAN was a central design idea underlying the development of both AT and SVO. But, the model of computation in the semantics for each of these was adapted from general models underlying epistemic logics to reason about distributed computing. Their primary focus was not authentication or even cryptographic protocols generally. It is perhaps not surprising, therefore, that previous analysis showed AT and SVO computational models not easily compatible with those of NPA [Mea94, Syv98].

Perhaps what is needed is a model of computation that is more directly intended to represent authentication protocols. One such model is strand spaces [THG97, THG98b]. (See also [Gut] in this volume. Related to strands is the multiset rewriting (MSR) approach [CDL+].) Besides being a model specifically directed at this problem area and having a growing base of theoretical literature, it seems to fit somewhat naturally to NPA and similar tools, e.g., Athena [Son99]. The question that naturally arises is then whether we can effectively repeat the above NPATRL idea using something like BAN for the requirements language and strands as the model. In other words, could we have a process as expressed in the following diagram?



An affirmative answer would require a strand semantics for a BAN-style language. We will present a proposal for one below. We shall first provide a brief overview of the relevant strand space concepts.

Overview of Strands

A strand is basically a local history of sent and received messages in a protocol run. A strand space is a collection of strands, and a bundle is a graph that reflects a causally meaningful way that a set of strands might be connected.

The messages sent between principals are taken from an algebra A of terms. We will say more about the algebra shortly. Terms can be signed, e.g., $+t$ or $-t$, to indicate sending and receiving of messages respectively. We will give definitions for all the relevant concepts below. First, here is a picture of a bundle for Protocol 7, the (abridged) Needham-Schroeder Public-Key Protocol.

$$\begin{array}{ccc} +\{n_A, A\}_{k_B} & \longrightarrow & -\{n_A, A\}_{k_B} \\ \Downarrow & & \Downarrow \\ -\{n_A, n_B\}_{k_A} & \longrightarrow & +\{n_A, n_B\}_{k_A} \\ \Downarrow & & \\ +\{n_B\}_{k_B} & & \end{array}$$

The vertical sequences of double arrows are the strands, the local traces of messages sent to and from a given principal (in a given run). The horizontal (single) arrows link one strand to another by connecting the transmission and the reception of the same message. We now give more precise definitions, all of which are taken from [THG99b].

Let Σ be a set of strands and $(\pm\mathsf{A})^*$ be the set of all finite sequences of signed terms. A *strand space* over A is a set Σ together with a trace mapping $tr : \Sigma \to (\pm\mathsf{A})^*$.

Fix a strand space Σ

1. A *node* is a pair $\langle s, i\rangle$, with $s \in \Sigma$ and i an integer satisying $1 \leq i \leq \text{length}(\text{tr}(s))$. The set of nodes is denoted by $\mathcal{N}$. We will say the node $\langle s, i\rangle$ belongs to the strand s. Clearly, every node belongs to a unique strand.
2. If $n = \langle s, i\rangle \in \mathcal{N}$ then $\text{index}(n) = i$ and $\text{strand}(n) = s$. Define $\text{term}(n)$ to be $(\text{tr}(s))_i$, i.e. the ith signed term in the trace of s. Similarly, $\text{uns\_term}(n)$ is $((\text{tr}(s))_i)_2$, i.e. the unsigned part of the ith signed term in the trace of s.
3. There is an edge $n_1 \to n_2$ if and only if $\text{term}(n_1) = +a$ and $\text{term}(n_2) = -a$ for some $a \in \mathsf{A}$. Intuitively, the edge means that node n_1 sends the message a, which is received by n_2, recording a potential causal link between those strands.
4. When $n_1 = \langle s, i\rangle$ and $n_2 = \langle s, i{+}1\rangle$ are members of $\mathcal{N}$, there is an edge $n_1 \Rightarrow n_2$. Intuitively, the edge expresses that n_1 is an immediate causal predecessor of n_2 on the strand s. We write $n' \Rightarrow^+ n$ to mean that n' precedes n (not necessarily immediately) on the same strand.
5. $\mathcal{N}$ together with both sets of edges $n_1 \to n_2$ and $n_1 \Rightarrow n_2$ is a directed graph $\langle \mathcal{N}, (\to \cup \Rightarrow)\rangle$.

Suppose $\to_{\mathcal{C}} \subseteq \to$; suppose $\Rightarrow_{\mathcal{C}} \subseteq \Rightarrow$; and suppose $\mathcal{C} = \langle \mathcal{N}_{\mathcal{C}}, (\to_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}})\rangle$ is a subgraph of $\langle \mathcal{N}, (\to \cup \Rightarrow)\rangle$. $\mathcal{C}$ is a *bundle* if:

1. $\mathcal{C}$ is finite.
2. If $n_2 \in \mathcal{N}_{\mathcal{C}}$ and $\text{term}(n_2)$ is negative, then there is a unique $n_1 \in \mathcal{N}_{\mathcal{C}}$ such that $n_1 \to_{\mathcal{C}} n_2$.
3. If $n_2 \in \mathcal{N}_{\mathcal{C}}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_{\mathcal{C}} n_2$.
4. $\mathcal{C}$ is acyclic.

If $\mathcal{S}$ is a set of edges, i.e. $\mathcal{S} \subseteq (\to \cup \Rightarrow)$, then $\prec_{\mathcal{S}}$ is the transitive closure of $\mathcal{S}$, and $\preceq_{\mathcal{S}}$ is the reflexive and transitive closure of $\mathcal{S}$. The relations $\prec_{\mathcal{S}}$ and

$\preceq_{\mathcal{S}}$ are each subsets of $\mathcal{N}_{\mathcal{S}} \times \mathcal{N}_{\mathcal{S}}$, where $\mathcal{N}_{\mathcal{S}}$ is the set of nodes incident with any edge in $\mathcal{S}$.

These are all of the definitions that we need to set out a possible worlds model and semantics for sending, receiving, and knowledge. We will provide below more details about the term algebra that will allow us to express, e.g., that a principal who receives a ciphertext (encrypted message) and has the decryption key has also got the unencrypted message.

Possible Worlds from Strand Spaces

We now describe a possible world semantics of epistemic logics for distributed computing in general and for security protocols in particular, for example, as presented in [AT91, SvO94, SvO96].

In a traditional system model and knowledge semantics for distributed computing, 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 modeling 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.

A run is a sequence of global states indexed by integers to represent time. 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 determines a possible world (sometimes also called nodes or points). We assume that global states are unique wrt runs and times. Thus, they can be referred to by, e.g., '$\langle r, t\rangle$'. At any given global state, various things will be true, e.g., that principal Q has previously sent the message $\{X\}_k$. What a principal P then knows (believes) at a given point $\langle r, t\rangle$ is precisely that which is true at all possible worlds with the same local state $r_P(t)$ for P as $\langle r, t\rangle$. This is typically captured by means of an accessibility relation on global states $\rightsquigarrow_P$ for a principal P. When the relation is an equivalence, it is also called an indistinguishability relation $\sim_P$ for a principal P. This allows for a simple intuitive definition, without even having to describe in any way properties of local states, viz:

- $\langle r, t\rangle \sim_P \langle r', t'\rangle$ iff P is in the same local state at both points, i.e., $r_P(t) = r'_P(t')$.

Given an indistinguishability relation, we can then go on to define principal P's knowledge in terms of the worlds that are P-indistinguishable.

- $\langle r, t\rangle \models P$ *knows* φ iff $\langle r', t'\rangle \models \varphi$ for all $\langle r', t'\rangle$ such that $\langle r, t\rangle \sim_P \langle r', t'\rangle$

The above system model and characterization of knowledge (belief) is essentially what is found in [AT91, SvO94, SvO96]. It is largely based on similar models and characterizations of knowledge in distributed computing; see for example [FHMV95]. Note that the relation just given is an equivalence relation, as is the strand-based relation to be given presently. For this reason, and to be

consistent with earlier literature such as [FHMV95], we refer to the associated modality as knowledge rather than belief, but no great significance should be attached to this choice, as we saw in Section 3.3. We now turn specifically to strand spaces as a basis for knowledge semantics.

Strand Semantics for Knowledge

In the conclusion of [THG97] it was suggested that,

> "[what] a protocol participant knows, in virtue of his experience in executing a protocol, is that he has performed the actions lying on some strand s. Thus, the real world must include some bundle $\mathcal{C}$ such that s is contained in $\mathcal{C}$. The beliefs that the participant may justifiably hold are those that are true in every bundle $\mathcal{C}$ containing s." [THG97]

Thus, a possible world on this approach is simply a bundle. This is a reasonable approach for reasoning about some protocol features. However, we found it also worthwhile to include in the definition of possible worlds the nodes within bundles. We did this in order to capture temporal aspects of the above authentication logics, specifically freshness. This will also facilitate the addition of richer temporal formulae to the logic, as in [Syv93a].

Neither strand spaces nor bundles have a notion of global time. Thus we cannot have an indistinguishability relation that corresponds directly to the above. However, $\langle \mathcal{C}, s, i\rangle$ picks a unique point $\langle s, i\rangle$ in bundle $\mathcal{C}$ and partitions $\mathcal{N}_\mathcal{C}$ into $\{\langle t, j\rangle : \langle t, j\rangle \preceq_\mathcal{C} \langle s, i\rangle\}$ and $\{\langle t, j\rangle : \langle t, j\rangle \not\preceq_\mathcal{C} \langle s, i\rangle\}$. This partition allows us to define an accessibility relation on nodes in bundles based on local time.

1. Given a strand s, let princ(s) refer to the principal whose strand s is.
2. Given a node $\langle s, i\rangle$ and a strand t in a bundle $\mathcal{C}$, let the *restriction of* t *to* $\langle s, i\rangle$ *in* $\mathcal{C}$ be $\mathrm{tr}(t) \restriction \langle s, i\rangle = \langle \mathrm{tr}(t)_1, \ldots, \mathrm{tr}(t)_j\rangle$, where $\langle t, j\rangle$ is the greatest node on t s.t. $\langle t, j\rangle \preceq_\mathcal{C} \langle s, i\rangle$.

With this notation in place we can now define an indistinguishability relation. Assume bundles $\mathcal{C}, \mathcal{C}'$, and strands s, s', and indices i, i' such that $\langle s, i\rangle \in \mathcal{N}_\mathcal{C}$ and, $\langle s', i'\rangle \in \mathcal{N}_{\mathcal{C}'}$. A natural definition, analogous to the runs-and-times definition of the traditional literature would be to have $\langle \mathcal{C}, s, i\rangle \sim_P \langle \mathcal{C}', s', i'\rangle$ (i.e., $\langle \mathcal{C}, s, i\rangle$ is *P-indistinguishable* from $\langle \mathcal{C}', s', i'\rangle$) just in case P's history in $\mathcal{C}$ up to $\langle s, i\rangle$ matches P's history in $\mathcal{C}'$ up to $\langle s', i'\rangle$. This is exactly right. However, just as there is no global time in a bundle, there may also be multiple strands associated with one principal. The resulting definition is thus:

$\langle \mathcal{C}, s, i\rangle$ is *P-indistinguishable* from $\langle \mathcal{C}', s', i'\rangle$ (written as $\langle \mathcal{C}, s, i\rangle \sim_P \langle \mathcal{C}', s', i'\rangle$) iff

1. for any t in $\mathcal{C}$ s.t. princ(t) = P there exists t' in $\mathcal{C}'$ s.t. $\mathrm{tr}(t) \restriction \langle s, i\rangle = \mathrm{tr}(t') \restriction (s', i')$ and princ(t') = P, and
2. the number of strands satisfying clause 1 is the same in $\mathcal{C}$ and $\mathcal{C}'$.

Truth Conditions for BAN-Style Formulae

The purpose of this section, is to present truth conditions for basic formulae of a BAN-style language. The basic notions we cover are freshness, key goodness, said and received (got) messages, and jurisdiction.

Given our definition of $\sim_P$ above we can now present truth conditions for knowledge in this semantics. Let φ be some formula in our language. We will define $\models$ inductively; however the presentation is organized pedagogically rather than to respect the inductive construction. We assume the usual truth conditions for logical connectives; although we will not discuss compound formulae here.

$$\langle \mathcal{C}, s, i \rangle \models P \textit{ knows } \varphi$$

iff $\langle \mathcal{C}', s', i' \rangle \models \varphi$ at all $\langle \mathcal{C}', s', i' \rangle$ s.t. $\langle \mathcal{C}, s, i \rangle \sim_P \langle \mathcal{C}', s', i' \rangle$

This definition gives a strand semantics for knowledge in a distributed environment. However, we have not yet described what specific types of things φ might express. Giving truth conditions for the various possibilities is the focus of the remainder of this section.

We can give semantics for formulae expressing the sending and receiving of messages without giving any more details about the model. Let M be an arbitrary message from our term algebra A. Then,

$$\langle \mathcal{C}, s, i \rangle \models P \textit{ sent } M$$

iff there is a node $\langle t, j \rangle$ in $\mathcal{C}$ s.t. (i) $\mathrm{princ}(t) = P$, (ii) $\langle t, j \rangle \preceq \langle s, i \rangle$, and (iii) $\mathrm{term}(\langle t, j \rangle) = +M$. Moreover,

$$\langle \mathcal{C}, s, i \rangle \models P \textit{ received } M$$

iff there is a node $\langle t, j \rangle$ in $\mathcal{C}$ s.t. (i) $\mathrm{princ}(t) = P$, (ii) $\langle t, j \rangle \preceq \langle s, i \rangle$, and (iii) $\mathrm{term}(\langle t, j \rangle) = -M$.

To give the truth conditions for other formulae, we must first spell out some of the structure of the term algebra and define a notion of submessage. The following definitions are taken from [THG99b] and can also be found in the preceding strand space papers.

Assume the following:

- A set $\mathbf{T} \subseteq \mathsf{A}$ of texts (representing the atomic messages).
- A set $\mathbf{K} \subseteq \mathsf{A}$ of cryptographic keys disjoint from $\mathbf{T}$, equipped with a unary operator $\mathsf{inv} : \mathbf{K} \to \mathbf{K}$.
  inv is injective; i.e., that it maps each member of a key pair for an asymmetric cryptosystem to the other; and that it maps a symmetric key to itself.
- Two binary operators

$$\mathsf{encr} : \mathbf{K} \times \mathsf{A} \to \mathsf{A}$$
$$\mathsf{join} : \mathsf{A} \times \mathsf{A} \to \mathsf{A}$$

We will follow notational conventions, some of which have already been mentioned, and write $\mathsf{inv}(k)$ as k^{-1}, $\mathsf{encr}(k, M)$ as $\{M\}_k$, and $\mathsf{join}(a, b)$ as $(a\, b)$. If K is a set of keys, K^{-1} denotes the set of inverses of elements of K.

The next assumption we make is that A is the algebra freely generated from $\mathbf{T}$ and $\mathbf{K}$ by the two operators encr and join. As noted in [THG99b], this assumption has been commonly made in this area of research going back to [DY83]. As in [THG99b] it is probably stronger than what we ultimately need but is pedagogically convenient. Amongst other things, it implies that encryptions and concatenations are unique and always distinct from each other and from $\mathbf{T}$ and $\mathbf{K}$.

Central to the semantics of *said* formulae is the concept of an ideal. Interestingly, in the strand space papers, it was introduced to formulate general facts about the penetrator's capabilities; while, for this discussion, we will say virtually nothing about the nature of the penetrator.

If $K \subseteq \mathbf{K}$, a K-ideal of A is a subset I of A such that for all $h \in I$, $g \in \mathsf{A}$ and $k \in K$

1. $h\, g, g\, h \in I$.
2. $\{h\}_k \in I$.

The smallest K-ideal containing h is denoted $I_K[h]$.

The notion of ideal can be used to define a subterm relation $\sqsubset$ as follows [THG98a].

Let $K \subseteq \mathbf{K}$. $s \in \mathsf{A}$ is a *K-subterm of* $t \in \mathsf{A}$, $(s \sqsubset_K t)$ iff $t \in I_K[s]$.

If $K = \mathbf{K}$ in this definition, then we say simply that *s is a subterm of t*, and write $s \sqsubset t$.

We now give truth conditions for *said* formulae

$$\langle \mathcal{C}, s, i \rangle \models P \textit{ said } M$$

iff there is a message M' s.t. $\langle \mathcal{C}, s, i \rangle \models P \textit{ sent } M'$ and $M \sqsubset_K M'$ where K is the set of keys possessed by P at $\langle s, i \rangle$.

Notice that P is held accountable, e.g., for saying M at n, if he sends $\{M\}_k$ at $n' \preceq n$ and he has k at n, even if k was not in his key set until some n'' s.t. $n' \prec n'' \preceq n$.

A definition that does not occur in any of the strand space papers is that of a filter. In many contexts, filters are the duals of ideals. In our case, they are useful for giving semantics to *got* formulae, those that express the understood messages contained in received messages. (Millen and Rueß introduce the same idea in [MR00] to reason about secrecy invariants. They call it a "coideal".)

If $K \subseteq \mathbf{K}$, a K-filter of A is a subset F of A such that for all $h, g \in \mathsf{A}$ and $k \in K$

1. $h\, g \in F$ implies $h \in F$ and $g \in F$
2. $\{h\}_k \in F$ implies $h \in F$ for $k^{-1} \in K$

The smallest K-filter containing h is denoted $F_K[h]$.

In general, the relation between filters and ideals is not so simple because, in public-key cryptography, one may have k and not have k^{-1}, or vice versa.

However, in this section we are limiting discussion to the symmetric key case, $k = k^{-1}$—for which there is a simple relation. (This relation also holds when both cognates of a public/private key pair are known.) It is easy to show that

Claim 4 *For all sets of keys K' of the form $K \cup K^{-1}$*

$$g \in F_{K'}[h] \text{ iff } h \in I_{K'}[g].$$

Thus, for key sets K' of this form, by definition 6.2, $s \sqsubset_{K'} t$ iff $s \in F_{K'}[t]$. We can now give the truth conditions for *got* formulae. (We present them for the general case.)

$$\langle \mathcal{C}, s, i \rangle \models P \textit{ got } M$$

iff there is a message M' s.t. $\langle \mathcal{C}, s, i \rangle \models P \textit{ received } M'$ and $M \in F_K[M']$ where K is the set of keys possessed by P at $\langle s, i \rangle$.

We can use the truth conditions for *said* and *got* formulae to further give the truth conditions for key goodness.

$$\langle \mathcal{C}, s, i \rangle \models P \overset{k}{\longleftrightarrow} Q$$

iff, for all $\langle s', i' \rangle \in \mathcal{N}_\mathcal{C}$, $\langle \mathcal{C}, s', i' \rangle \models R \textit{ said } \{M \textit{ from } Q\}_k$ implies either $\langle \mathcal{C}, s', i' \rangle \models R \textit{ received } \{M \textit{ from } Q\}_k$, or $R = Q$ and $\langle \mathcal{C}, s', i' \rangle \models R \textit{ said } M$. Moreover, if $\langle \mathcal{C}, s', i' \rangle \models R \textit{ said } \{M\}_k$ (instead of the stronger $\langle \mathcal{C}, s', i' \rangle \models R \textit{ said } \{M \textit{ from } Q\}_k$), then $R \in \{P, Q\}$ (instead of the stronger $R = P$).

Note that these are the truth conditions from [SvO96] with $\langle \mathcal{C}, s, i \rangle$ replacing $\langle r, t \rangle$ and $\langle \mathcal{C}, s', i' \rangle$ replacing $\langle r, t' \rangle$ throughout. This was itself based on the truth conditions for goodness given in [AT91].

Once we have a mechanism to express the beginning of the current epoch, we will be able to similarly dispatch the freshness and jurisdiction formulae. In order to do that, we must again confront the absence of a global concept of time. In the system models for possible world semantics of BAN-like logics, it was trivial to stipulate a global time t_0 and then define something as fresh if it was not said (by anyone) prior to t_0. We instead define a concept now as follows.

For any bundle $\mathcal{C}$, $\mathsf{now}_\mathcal{C} \subseteq \mathcal{N}_\mathcal{C}$, is a nonempty set of incomparable nodes (i.e., a nonempty set of nodes s.t. $n, n' \in \mathsf{now}_\mathcal{C}$ implies $n \not\preceq n'$ and $n' \not\preceq n$). For $n \in \mathcal{N}_\mathcal{C}$, we may write '$\mathsf{now}_\mathcal{C} \preceq n$' just in case there exists $n' \in \mathsf{now}_\mathcal{C}$ s.t. $n' \preceq n$. When it is clear from context which bundle is relevant, we will write simply 'now'.

Thus,

$$\langle \mathcal{C}, s, i \rangle \models \textit{fresh}(M)$$

iff for all principals P, $\langle \mathcal{C}, s', i' \rangle \models P \textit{ said } M$ implies $\mathsf{now} \preceq \langle s', i' \rangle$.

The truth conditions for jurisdiction assume truth conditions for *says* formulae, which the definition of $\mathsf{now}_\mathcal{C}$ allows us to formulate.

$$\langle \mathcal{C}, s, i \rangle \models P \textit{ says } M$$

iff there is a message M' and a node $\langle t, j\rangle$ in $\mathcal{C}$ s.t. (i) $\mathrm{princ}(t) = P$, (ii) $\mathsf{now} \preceq \langle t, j\rangle \preceq \langle s, i\rangle$, (iii) $\mathrm{term}(\langle t, j\rangle) = +M'$, and (iv) $M \sqsubset_K M'$ where K is the key set possessed by P at $\langle s, i\rangle$.

If φ is a formula.

$$\langle \mathcal{C}, s, i\rangle \models P \textit{ controls } \varphi$$

iff $\langle \mathcal{C}, s, i\rangle \models P \textit{ says } \varphi$ implies $\langle \mathcal{C}, s', i'\rangle \models \varphi$ for any $\langle s', i'\rangle$ s.t. $\mathsf{now} \preceq \langle s', i'\rangle$.

These conditions are similar to those in [AT91] and [SvO94, SvO96], *mutatis mutandis*. Notice that goodness is a condition that is constant across all points in the same bundle. And, jurisdiction and freshness are constant across all points in the present epoch. Notice also that jurisdiction is restricted to those messages that are formulae, rather than messages in general.

This completes our presentation of truth conditions. Strand based truth conditions for public keys, Diffie-Hellman, and other aspects of SVO have yet to be developed. What we have done is to provide a means by which BAN-style requirements can be mapped to strand-style protocol specifications. Something like this is necessary for the protocol analysis approach characterized by the diagram at the beginning of this section. For the "strand machine" to process its inputs there must be some means for it to combine them. The mapping provides such a means. To completely develop the semantic approach using BAN-style requirements for a strand-style model, the strand machine itself must be built. We conclude with a description of some of other areas where there is still much work to be done.

7 The Future

In [Mea00b], Meadows sets out a number of open areas in the application of formal methods to cryptographic protocols. The primary focus is beyond simple two-party authentication protocols, and that paper is a good place to get an idea of where much of the cutting edge research is or soon will be. We finish up with a discussion of these open areas, but with a slant towards the kinds of formalisms and ideas that have been discussed above. We also try to mention some of the recent work which has not been alluded to elsewhere above. Indeed, such a large amount of work has been done in formal analysis of authentication and other security protocols that, despite the number of references cited herein, far more work has gone unmentioned, much of it quite good.

Appropriately, one of the open areas is in *open-ended protocols*. The two major ways in which a protocol can be open-ended is in what data is sent and in who is sending or receiving. We address these in order.

A protocol may be open-ended in virtue of the data sent. For example, the Internet Key Exchange Protocol (IKE) that is part of IPSEC [DH99], includes an agreement on a Security Association (SA). The SA includes such things as a choice of algorithms and other parameters. But, there is no defined (upper) limit on what can be included in an SA. This sort of open-endedness has not been formally analyzed as far as we know. Another aspect of IKE is that the SA has a more elaborate, indeed open-ended, data structure than a simple cryptographic

key. In classic authentication protocols, the data about which we prove authentication and secrecy properties is simply a key. In Diffie-Hellman exchange, there may be parts contributed by the principals that make up the key, but Diffie-Hellman based protocols have been analyzed using VO and SVO [vO93, SvO96].

Protocols can also be open-ended in the participants involved. An obvious example is in various kinds of group protocols. These can be for both group authentication and group confidentiality properties. One example of such a group protocol is a group signature protocol, in which a signature can identify only that the signer was from a group unless an additional protocol is run (typically with a trusted authority) to reveal the individual responsible for the given signature. As introduced in [CvH91], these were perhaps only open-ended in principle since there was no efficient means to add members to an existing group. The first significant advance on that problem was made in [CS97], and others have since followed. There has not been any formal methods work that we know of directly on this area. More positive results have been seen in the area of group Diffie-Hellman [AST98]. These are essentially Diffie-Hellman type establishments for open-ended groups. Meadows was able to analyze these protocols after expanding NPA [Mea00a]. More recently, Meadows has presented evaluations of secure multicast to the IETF and the IRTF Secure Multicast Group. We have begun specification and examination of secure multicast protocols using NPA-TRL. Other formal methods work involving groups of arbitrary size can be found in [Pau97, BS97]. Both of these papers make use of theorem proving, Isabelle and PVS respectively to examine the same protocol.

Another important open are is *denial of service*. Meadows has devised a framework [Mea01] for reasoning about denial of service in cryptographic protocols, although not a formal method per se. The problem with authentication is that it is not only a protection against but a great source of denial-of-service attacks. If only authenticated principals are allowed to perform any actions, then unauthenticated principals cannot deny service. But, verifying authentication typically involves computationally intensive cryptographic operations. Thus, initiating many authentic connections can be an even more effective denial-of-service attack than simply initiating many connections. Meadows builds on the fail-stop concept set out in Section 5.3. The idea is to have the amount of work expended to defend a protocol against denial of service increase as the protocol progresses. The protocol is analyzed to show that it is fail-stop against an attacker whose capabilities are within a specified constraint. Note that this is a diversion from the Dolev-Yao intruder model that we have assumed throughout, up to this point. Obviously a Dolev-Yao intruder can arbitrarily deny service. Much of the open work involves backing off from such an unrealistically strong attacker to consider properties that can be established in the face of a different attacker.

Electronic commerce, in particular non-repudiation and fair exchange, is an area that has seen an explosion of protocols and also some formal work in the last several years. In fair exchange, there is no adversary per se. Rather, the idea is to make sure that each party gets his goods, signed contract, etc. just in

case the other does as well. In non-repudiation, the goal is to have evidence that a principal cannot repudiate. This can be evidence of messages sent (evidence of origin) or messages received (evidence of receipt). Obviously fair exchange and non-repudiation are closely related. The first attempt to reason about this area formally was by Kailar using a BAN-like logic [Kai95, Kai96]. The central logical construct is CanProve as in "A CanProve B says X". Zhou and Gollman also used SVO to reason about non-repudiation properties [ZG98]. We have already mentioned Brackin's verification of the Cybercash main sequence protocol using BGNY [Bra97]. A more recent approach to non-repudiation, using temporal logic with a game semantics can be found in [KR00].

The SET protocol is a good illustrator of several of the complexities we have introduced in this section. Like IKE, it is not a single protocol but a collection of subprotocols. As mentioned in Section 6.1, the protocol is very large and complex with many options, yet its specification lacks even an informal statement of requirements. It has a more elaborate structure than just a key on which principals must agree: there is a transaction on which the customer, merchant, and bank must agree, but parts of the transaction are hidden from some of the principals and parts are added to it as the protocol progresses. And, the reason that parts of the transaction are hidden is because the principals are mutually mistrusting and attempting some sort of non-repudiable fair exchange. Nonetheless, NPATRL was adapted to express requirements for payments in SET and related protocols by adding abstract structures for which some of the components are not revealed [MS98]. Also, the cardholder registration subprotocol has been verified using Isabelle and HOL [BMPT00]. Recall that in SVO and the AUTLOG based logic of [WK96] one can reason about principals' beliefs concerning messages in which not all the parts are recognizable. This would seem naturally generalizable to SET. In [KN98], Kessler and Neuman devised a logic for reasoning about payment in SET that combine elements from these logics, from Kailar's logic of accountability, and from the Stubblebine-Wright logic of recent security [SW96].

These large protocol suites raise still another open issue: *protocol composability*. The fail-stop protocols of Section 5.3 constitute one answer to this problem. But are there less onerous design restrictions that can be imposed (similar constraints on composition are given in [HT96])? It might seem that protocol composability is completely guaranteed by having only EFS protocols. However, even when the protocols are all EFS, the application environment generally will not be. Thus, there are still oracles available for active attacks.

Suppose that principals are willing to use keys obtained through a key-distribution protocol before the protocol completes. This is sometimes called "eager" use of keys in the literature. Only if the authentication protocol does not complete within some reasonable timeout is there an alarm or noting of anomaly in the logs. This eagerness might be all the more reasonable if the protocol distributing the keys is EFS. In this case, there would seem to be no possibility of mistake about who the session key is for, who the relevant principals are, or the roles they each play (i.e., initiator or responder). But, allowing

eager use of keys in an application that authenticates a random challenge by encryption using the session key could be used to attack the protocol. (This could be a variant of the sensor example of Protocol 6.)

Specifically, suppose Alice begins NSSK (Protocol 1) for a session with Bob, the attacker prevents the third message from arriving. Then, for the application challenge-response he produces:

Application Message 1 $E_B \rightarrow A : n_B$

Application Message 2 $A \rightarrow E_B : \{n_B\}_{K_{ab}}$

The attacker uses the response from Alice for the fourth message in NSSK, and intercepts the final message from Alice to Bob. Alice will now be spoofed into thinking she has completed a handshake with Bob when Bob was never present.

This attack is even possible if NSSK is strengthened to be made EFS. The point is to show that the applications that use keys established in an authentication protocol must also be considered. This aspect of protocol composability has received only a little attention. A version of this attack and related issues are discussed in [CMS01]. Besides general composable protocol design, there has also been a little work done into showing that particular protocols are composable [Mea99a, THG99a].

Another type of composability is between protocols and the cryptographic algorithms they employ. Protocol analysis as we have described it herein has treated cryptography as a black box, but some protocols and algorithms are secure if used in one combination while they are insecure in different combinations. Formal work going beyond black box treatments of cryptography in protocol analysis is just beginning [AR00, Can00, Jür00].

We mentioned the inappropriateness of Dolev-Yao adversaries for modeling denial-of-service attacks. They are also clearly inadequate for exchange protocols involving mutually mistrusting parties. Another area in which a Dolev-Yao adversary is simply too strong is anonymity. *Anonymity* services that have either been designed to be practical for most applications or that have actually been fielded are simply broken against a Dolev-Yao adversary [Oni, Ano, Cro, Fre]. One reason is that anonymity for all of these involves passing messages through an intermediate point so as to obscure identity of an originator from anyone observing a transmission. Some involve hopping through several points and some change the appearance of messages at each point so that parts of the transmission cannot be compared and seen to *be* parts of the same transmission. No matter how many of these precautions are taken, in a system where all messages pass through the intruder, the intruder will know exactly who is talking to whom (and possibly what is being said unless confidentiality is also protected). There are communication mechanisms that are secure against a Dolev-Yao intruder, e.g., dining cryptographer (DC) nets [Cha88]. However, nothing that is practical for widely used email, Web browsing, remote login, etc. is secure against a Dolev-Yao intruder. In [SS99], an epistemic model and logic was introduced for reasoning about group principals. This built on ideas in [FHMV95]. Recall from Section 6.2 that the usual model of computation associated with these logics has a single principal to represent the environment/penetrator. This is in

perfect keeping with the Dolev-Yao model. However, in [SS99], all communication principals, including the environment must be specified. And, there is in fact no single environment. Rather, there are many environment principals that have various capabilities and properties and that can be assembled in a variety of ways, i.e., into various sorts of group principals. One can then reason about various properties associated with a group of principals (the 'good guys') that another group of principals (the intruder) can actively or passively determine. For example, a particular distributed intruder may be able to determine that some (atomic) subprincipal of a group principal of cardinality n was the source of a message, but cannot narrow the cardinality lower than n. Work is underway to combine this approach, which has an intuitive yet formal expressiveness, with a CSP based approach [SS96]. The intent is to use the CSP as a semantics, much as the strand semantics for BAN described in Section 6.2. The language in [SS99] includes threshold-group principals and other primitives that should make it applicable to other areas besides anonymity.

Childhood's End

Specification and analysis of basic authentication protocols has been the focus of much of the above discussion—and much of the work in the last dozen years of formal methods in application to cryptographic protocols. The main concepts have been extensively explored and both intuitive and fully automated techniques have been developed, techniques that now do a thorough job and require no great sophistication. It has been several years since merely documenting a new attack on such protocols or devising a new formal method for reasoning about them was sufficient for publication in even small workshops. This is a positive sign. More complex protocols and protocols to accomplish more ambitious and subtle goals continue to come along. Formal methods are increasingly employed in the specification and analysis of protocols that are more than academic exercises: commercial products, complex protocol suites, international standards, etc. And, they have begun to have an impact in the real-world protocols that are being deployed. At the same time there has been a resurgence in theoretical models of both the new and the classic concepts, and these have in turn influenced the development and refinement of formal methods for protocol analysis and even design. It's an exciting time to be in the field.

References

[ABKL90] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart-cards. Research Report 67, Digital Systems Research Center, October 1990. Revised July, 1992. 69

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, April 1997. 130

[AG99] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 143:1–70, 1999. An extended

version of this paper appears as Research Report 149, Digital Equipment Corporation Systems Research Center, January 1998. An early presentation appears in [AG97]. 113

[AN94] Martín Abadi and Roger Needham. Prudent engineering practices for cryptographic protocols. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 122–136. IEEE CS Press, May 1994. 106, 130

[AN95] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO '95*, pages 236–247. Springer-Verlag, LNCS 963, August 1995. 107, 109

[AN96] Martín Abadi and Roger Needham. Prudent engineering practices for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996. A preliminary version appeared as [AN94]. 106

[Ano] The anonymizer. `http://www.anonymizer.com/`. 128

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptographic protocols (the compuational soundness of formal encryption. In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*. Springer-Verlag, LNCS, 2000. 128

[AST98] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik. Authenticated group key agreement and friends. In 5^{th} *ACM Conference on Computer and Communications Security (CCS'98)*, pages 17–26. ACM Press, November 1998. 126

[AT91] Martín Abadi and Mark R. Tuttle. A semantics for a logic of authentication. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, August 1991. 67, 76, 79, 88, 95, 120, 124, 125

[BAN88] Michael Burrows, Martín Abadi, and Roger Needham. Authentication: A practical study of belief in action. In M. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge (Tark)*, pages 325–342. Morgan Kaufmann, March 1988. Also presented at The Computer Security Foundations Workshop, Franconia, NH, June 1988. 66

[BAN89a] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. Research Report 39, Digital Systems Research Center, February 1989. Revised Feb. 22, 1990. 63, 66, 67, 68, 69, 70, 85, 87, 93, 95, 98, 101, 102

[BAN89b] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Operating Systems Review*, 23(5):1–13, December 1989. This issue of *OSR*: Proceedings of the Twelfth ACM Symposium on Operating Systems Principles (SOSP), Litchfield Park, Arizona, December 1989. 66

[BAN89c] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 426(1871):233–271, December 1989. 66

[BAN90a] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, Feb 1990. 66

[BAN90b] Michael Burrows, Martín Abadi, and Roger Needham. Rejoinder to nessett. *Operating Systems Review*, 24(2):39–40, April 1990. 75

[BAN90c] Michael Burrows, Martín Abadi, and Roger Needham. The scope of a logic of authentication. In J. Feigenbaum and M. Merritt, editors, *Distributed*

Computing and Cryptography, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 119–126. AMS and ACM, 1990. Proceedings of a DIMACS workshop, October 1989. 66

[Bie90] Pierre Bieber. A logic of communication in hostile environment. In *Proceedings of the Computer Security Foundations Workshop III*, pages 14–22. IEEE CS Press, June 1990. 97, 98

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84. IEEE CS Press, May 1992. 108

[BM93] Steve Bellovin and Michael Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 244–250. ACM Press, November 1993. 108

[BMPT00] Giampaolo Bella, Fabio Massacci, Lawrence C. Paulson, and Piero Tramontano. Formal verification of cardholder registration in SET. In F. Cuppens, Y. Deswarte, D. Gollmann, and M. Waidner, editors, *Computer Security – ESORICS 2000*, pages 159–174. Springer-Verlag, LNCS 1895, October 2000. 127

[Bra96] Stephen H. Brackin. A HOL extension of GNY for automatically analyzing cryptographic protocols. In *9th IEEE Computer Security Foundations Workshop*, pages 62–76. IEEE CS Press, June 1996. 112

[Bra97] Stephen H. Brackin. Automatic formal analyses of two large commercial protocols. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997. Available at `http://dimacs.rutgers.edu/Workshops/Security/program2/brackin.html`. 112, 127

[Bra98] Stephen H. Brackin. Evaluating and improving protocol analysis by automatic proof. In *11th IEEE Computer Security Foundations Workshop*, pages 138–152. IEEE CS Press, 1998. 112

[Bra00] Stephen H. Brackin. Automatically detecting most vulnerabilities in cryptographic protocols. In *DISCEX 2000: Proceedings of the DARPA Information Survivability Conference and Exposition*, volume I, pages 222–236. IEEE CS Press, January 2000. 112

[BS97] Jeremy Bryans and Steve Schneider. CSP, PVS and a recursive authentication protocol. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997. Available at `http://dimacs.rutgers.edu/Workshops/Security/program2/program.html`. 126

[BSW98] Levente Buttyán, Sebastian Staamann, and Uwe Wilhelm. A simple logic for authentication protocol design. In *11th IEEE Computer Security Foundations Workshop*, pages 153–162. IEEE CS Press, June 1998. 113

[Can00] Ran Canetti. A unified framework for analyzing security of protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/`. 128

[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, December 1993. 93

[Car94] Ulf Carlsen. Generating formal cryptographic protocol specifications. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 137–146. IEEE CS Press, May 1994. 98

[CDL+] Iliano Cervesato, Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. A comparison between strand spaces and transition systems for the specification of security protocols. To appear as a technical report, Department of Computer Science, Stanford University. 118

[Cha83] David Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology – Proceedings of Crypto 82*, pages 199–203, 1983. 107

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and receiver untraceability. *Journal of Cryptology*, 1(1):65–75, 1988. 128

[Che80] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980. 78, 79

[CJ97] John Clark and Jeremy Jacob. A survey of authentication protocol literature: Version 1.0, nov 1997. Available at `www-users.cs.york.ac.uk/~jac/` under the link "Security Protocols Review". 87, 93, 102, 103, 112

[CJ00] John Clark and Jeremy Jacob. Searching for a solution: Engineering tradeoffs and the evolution of provably secure protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 82–95. IEEE CS Press, May 2000. 113

[CMS01] Ran Canetti, Catherine Meadows, and Paul Syverson. Environmental requirements and authentication protocols. In *Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS)*, March 2001. A version of this paper has been invited for a special issue of *Requirements Engineering*. 128

[Cro] Crowds. `http://www.research.att.com/projects/crowds/`. 128

[CS97] Jan L. Camenisch and Markus A. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *Advances in Cryptology – CRYPTO '97*, pages 410–424. Springer-Verlag, LNCS 1294, 1997. 126

[CvH91] David Chaum and Eugène van Heyst. Group signature. In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, pages 257–265. Springer-Verlag, LNCS 547, 1991. 126

[Dek00] Anthony H. Dekker. C3po: a tool for automatic sound cryptographic protocol analysis. In *13th IEEE Computer Security Foundations Workshop*, pages 77–87. IEEE CS Press, June 2000. 83, 113

[DH99] Naganand Doraswamy and Dan Harkins. *IPSEC: The New Security Standard for the Internet, Intranets, and Virtual Private Networks*. Prentice Hall, 1999. 76, 125

[DM00] G. Denker and J. Millen. Capsl integrated protocol environment. In *DISCEX 2000: Proceedings of the DARPA Information Survivability Conference and Exposition*, volume I, pages 207–221. IEEE CS Press, January 2000. 112

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981. 73, 92

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, 2:107–125, 1992. 104

[DY83] Danny Dolev and Andrew C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, March 1983. 64, 123

[ES00] Neil Evans and Steve Schneider. Analysing time dependent security properties in CSP using PVS. In F. Cuppens, Y. Deswarte, D. Gollmann, and

M. Waidner, editors, *Computer Security – ESORICS 2000*, pages 222–237. Springer-Verlag, LNCS 1895, October 2000. 103

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995. 120, 121, 128

[Fre] Freedom. `http://www.freedom.net/`. 128

[GNY90] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, 1990. 76, 88, 89, 95

[Gol92] Robert Goldblatt. *Logics of Time and Computation,* 2^{nd} *edition*, volume 7 of *CSLI Lecture Notes*. CSLI Publications, 1992. 79

[Gol96] Dieter Gollmann. What do we mean by entity authentication? In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 46–54. IEEE CS Press, May 1996. 99

[Gol00] Dieter Gollmann. On the verification of cryptographic protocols - a tale of two committees. In Steve Schneider and Peter Ryan, editors, *Electronic Notes in Theoretical Computer Science*, volume 32. Elsevier Science Publishers, 2000. 102, 105

[GS98] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In R. K. Iyer, M. Morganti, W. K. Fuchs, and V. Gligor, editors, *Dependable Computing for Critical Applications 5*, pages 79–100. IEEE Computer Society Press, 1998. 109

[Gut] Joshua D. Guttman. Security goals: Packet trajectories and strand spaces. This volume. 113, 118

[HT96] N. Heintze and J. D. Tygar. A model for secure protocols and their composition. *IEEE Transactions on Software Engineering*, 22(1):16–30, January 1996. 127

[Jür00] Jan Jürjens. Bridging the gap: Formal vs. complexity-theoretical reasoning about cryptography, December 2000. Presentation at the Schloss Dagstuhl Seminar on Security through Analysis and Verification. 128

[Kai95] Rajashekar Kailar. Reasoning about accountability in protocols for electronic commerce. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 236–250. IEEE CS Press, May 1995. 127

[Kai96] Rajashekar Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on Software Engineering*, 5(22), May 1996. 127

[Kem87] Richard A. Kemmerer. Using formal verification techniques to analyze cryptographic protocols. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, pages 134–139. IEEE CS Press, May 1987. 63

[KN98] Volker Kessler and Heike Neumann. A sound logic for analysing electronic commerce protocols. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Computer Security – ESORICS 98*, pages 345–360. Springer-Verlag, LNCS 1485, September 1998. 127

[KR00] Steve Kremer and Jean-François Raskin. A game approach to the verification of exchange protocols: Application to non-repudiation protocols. In P. Degano, editor, *First Workshop on Issues in the Theory of Security – WITS'00*, pages 93–98, July 2000. 127

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978. 109

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. *Software - Concepts and Tools*, 17:93–102, 1996. 92, 101

[Low97] Gavin Lowe. A herarchy of authentication specifications. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW9)*, pages 31–43. IEEE Computer Society Press, June 1997. 100, 101, 102, 103

[MCF87] Jonathan K. Millen, Sidney C. Clark, and Sheryl B. Freedman. The Interrogator: Protocol security analysis. *IEEE Transactions on Software Engineering*, 13(2):274–288, 1987. 63

[Mea94] Catherine Meadows. A model of computation for the NRL Protocol Analyzer. In *Proceedings of the 7th Computer Security Foundations Workshop*, pages 84–89. IEEE CS Press, June 1994. 114, 116, 118

[Mea96] Catherine Meadows. The NRL Protocol Analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, February 1996. 114, 116

[Mea99a] C. Meadows. Analysis of the Internet Key Exchange protocol using the NRL Protocol Analyzer. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1999. 128

[Mea99b] Catherine Meadows. A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW12)*, pages 4–13. IEEE CS Press, June 1999. 134

[Mea00a] Catherine Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In P. Degano, editor, *First Workshop on Issues in the Theory of Security – WITS'00*, pages 87–92, July 2000. 126

[Mea00b] Catherine Meadows. Open issues in formal methods for cryptographic protocol analysis. In *DISCEX 2000: Proceedings of the DARPA Information Survivability Conference and Exposition*, volume I, pages 237–250. IEEE Computer Society Press, January 2000. 125

[Mea01] Catherine Meadows. A cost-based framework for analysis of denial of service in networks. *Journal of Computer Security*, 2001. Forthcoming. A preliminary version of portions of this work appeared in [Mea99b]. 126

[Men87] Elliott Mendelson. *Introduction to Mathematical Logic*. Wadsworth Publishing Co., 1987. 78

[Mil] J. Millen. Capsl Web site. `www.csl.sri.com/~millen/capsl`. 112

[Mil84] Jonathan K. Millen. The Interrogator: A tool for cryptographic protocol security. In *Proceedings of the 1984 IEEE Symposium on Security and Privacy*, pages 134–141, Oakland, CA, April 1984. IEEE Computer Society Press. 63

[Mos86] Paul K. Moser, editor. *Empirical Knowledge: Readings in Contemporary Epistemology*. Rowman & Littlefield, 1986. 79

[MR00] Jon Millen and Harald Rueß. Protocol-independent secrecy. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 110–119, May 2000. 123

[MS98] C. Meadows and P. Syverson. A formal specification of requirements for payment trnasactions in the SET protocol. In R. Hirschfeld, editor, *Financial Cryptography, FC'98*, pages 122–140. Springer-Verlag, LNCS 1465, 1998. 117, 127

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. 64, 80, 89

[Nes90] D. M. Nessett. A critique of the burrows, abadi, and needham logic. *Operating Systems Review*, 24(2):35–38, April 1990. 73, 74

[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, 1978. 65, 101, 102

[NS93] B. Clifford Neuman and Stuart G. Stubblebine. A Note on the Use of Timestamps as Nonces. *Operating Systems Review*, 27(2):10–14, April 1993. 93

[Oni] Onion routing. `http://www.onion-router.net/`. 128

[Pau97] Lawrence C. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop (CSFW10)*, pages 84–94. IEEE CS Press, June 1997. 126

[PS00] Adrian Perrig and Dawn Song. Looking for diamonds in the desert — extending automatic protocol generation to three-party authentication and key agreement. In *13th IEEE Computer Security Foundations Workshop*, pages 64–76. IEEE CS Press, June 2000. 113

[Ros96] A. W. Roscoe. Intensional specification of security protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop (CSFW9)*, pages 28–36. IEEE CS Press, June 1996. 103

[Sat89] M. Satyanarayanan. Integrating security in a large distributed system. *ACM Transactions on Computer Systems*, 15(3):247–280, August 1989. 102

[Sch96] Bruce Schneier. *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1996. 64, 80, 107

[SET97] Secure Electronic Transaction Specification, Version 1.0, May 1997. `http://www.visa.com/set/`. 117

[SM93] P. Syverson and C. Meadows. A logical language for specifying cryptographic protocol requirements. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 165–177. IEEE CS Press, May 1993. 117

[SM94] P. Syverson and C. Meadows. Formal requirements for key distribution protocols. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, pages 32–331. Springer-Verlag, LNCS 950, 1994. 117

[SM96] P. Syverson and C. Meadows. A formal language for cryptographic protocol requirements. *Designs, Codes, and Cryptography*, 7(1 and 2):27–59, January 1996. 114, 116, 117

[Sne91] Einar Snekkenes. Exploring the BAN approach to protocol analysis. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 171–181. IEEE CS Press, May 1991. 96, 97

[Sne92] Einar Snekkenes. Roles in cryptographic protocols. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 105–119. IEEE CS Press, May 1992. 98

[Son99] Dawn Song. Athena: a new efficient automatic checker for security protocol analysis. In *Proceedings of the Twelth IEEE Computer Security Foundations Workshop*, pages 192–202, Mordano, Italy, June 1999. IEEE Computer Society Press. 118

[SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In E. Bertino, H. Kurth, G. Martella, and E. Montolivio, editors, *Computer Security – ESORICS 96*, pages 198–218. Springer Verlag, LNCS 1146, 1996. 129

[SS99] Paul Syverson and Stuart Stubblebine. Group principals and the formalization of anonymity. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 – Formal Methods, Vol. I*, pages 814–833. Springer-Verlag, LNCS 1708, 1999. 128, 129

[Sti95] Douglas R. Stinson. *Cryptography: Theory and Practice*. CRC Press, 1995. 64

[SvO94] Paul F. Syverson and Paul C. van Oorschot. On unifying some cryptographic protocols. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 14–28. IEEE CS Press, May 1994. 77, 78, 120, 125

[SvO96] Paul F. Syverson and Paul C. van Oorschot. A unified cryptographic protocol logic. NRL Publication 5540-227, Naval Research Lab, 1996. 77, 78, 79, 84, 85, 88, 95, 98, 120, 124, 125, 126

[SW96] Stuart Stubblebine and Rebecca Wright. An authentication logic supporting synchronization, revocation, and recency. In *3rd ACM Conference on Computer and Communications Security (CCS'96)*, pages 95–105. ACM Press, March 1996. 127

[Syv91] Paul F. Syverson. The use of logic in the analysis of cryptographic protocols. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 156–170. IEEE CS Press, May 1991. 75, 76

[Syv92] Paul F. Syverson. Knowledge, belief, and semantics in the analysis of cryptographic protocols. *Journal of Computer Security*, 1(3,4):317–334, 1992. 76, 79

[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, November, 1993. ACM Press. 96, 98, 121

[Syv93b] Paul F. Syverson. On Key Distribution Protocols for Repeated Authentication. *Operating Systems Review*, 27(4):24–30, October 1993. 93

[Syv94] Paul Syverson. A taxonomy of replay attacks. In *Proceedings of the Computer Security Foundations Workshop (CSFW7)*, pages 187–191. IEEE CS Press, June 1994. 91, 92, 93, 95, 98

[Syv96] Paul Syverson. Limitations on design principles for public key protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 62–72. IEEE Computer Society Press, May 1996. 107, 108

[Syv98] Paul F. Syverson. Relating two models of computation for security protocols. In *Workshop on Formal Methods and Security Protocols*, Indianapolis, Indiana, June 1998. Available at `http://www.cs.bell-labs.com/who/nch/fmsp/program.html`. 118

[Syv00] Paul F. Syverson. Towards a strand semantics for authentication logic. *Electronic Notes in Theoretical Computer Science*, 20, 2000. Proceedings of MFPS XV (S. Brookes, A. Jung, M. Mislove and A. Scedrov, eds.), New Orleans, LA, April 1999. 114, 118

[THG97] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces. Technical report, The MITRE Corporation, November 1997. 66, 114, 118, 121

[THG98a] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Honest ideals on strand spaces. In *Proceedings of the 1998 IEEE Computer Security Foundations Workshop — CSFW'11*, pages 66–77. IEEE Computer Society Press, 1998. 123

[THG98b] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 160–171, Oakland, CA, May 1998. IEEE Computer Society Press. 66, 114, 118

[THG99a] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Mixed strand spaces. In P. Syverson, editor, *Proceedings of the 12th IEEE Computer Security Foundations Workshop — CSFW'99*, pages 72–82, Mordano, Italy, June 1999. IEEE Computer Society Press. 128

[THG99b] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2,3):191–230, 1999. 119, 122, 123

[Tou92] Marie-Jeanne Toussaint. Separating the specification and implementation phases in cryptology. In Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *Computer Security – ESORICS 92*, pages 77–102. Springer-Verlag, LNCS 648, November 1992. 108

[Tut] Mark Tuttle. Flaming in Franconia. Remarks made in a panel discussion on the use of formal methods in the analysis of cryptographic protocols at the IEEE Computer Security Foundations Workshop in Franconia, New Hampshire, June 1992. 79

[vO93] Paul C. van Oorschot. Extending cryptographic logics of belief to key agreement protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 233–243. ACM Press, November 1993. 67, 76, 85, 88, 95, 98, 126

[WK96] Gabriele Wedel and Volker Kessler. Formal semantics for authentication logics. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Computer Security – ESORICS 96*, pages 219–241. Springer-Verlag, LNCS 1146, September 1996. 78, 83, 113, 127

[Yah93] Raphael Yahalom. Optimality of asynchronous two-party secure data-exchange protocol. *Journal of Computer Security*, 2(2–3):191–209, 1993. 93

[ZG98] Jianying Zhou and Dieter Gollmann. Towards verification of non-repudiation protocols. In T. Vickers J. Grundy, M. Schwenke, editor, *International Refinement Workshop and Formal Methods Pacific 1998*, pages 370–380. Springer-Verlag, 1998. 127