Formal Methods for the Analysis of Authentication Protocols
CITI Technical Report 93-7

A. D. Rubin
P. Honeyman
Center for Information Technology Integration
Dept. of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48103-4943

November 8, 1993

Abstract

In this paper, we examine current approaches and the state of the art in the application of formal methods to the analysis of cryptographic protocols. We use Meadows' classification of analysis techniques into four types.

The Type I approach models and verifies a protocol using specification languages and verification tools not specifically developed for the analysis of cryptographic protocols. In the Type II approach, a protocol designer develops expert systems to create and examine different scenarios, from which she may draw conclusions about the security of the protocols being studied. The Type III approach models the requirements of a protocol family using logics developed specifically for the analysis of knowledge and belief. Finally, the Type IV approach develops a formal model based on the algebraic term-rewriting properties of cryptographic systems.

The majority of research and the most interesting results are in the Type III approach, including reasoning systems such as the BAN logic; we present these systems and compare their relative merits. While each approach has its benefits, no current method is able to provide a rigorous proof that a protocol is secure.

Contents

1 Introduction

Authentication is the process by which a principal in a distributed system proves its identity. Typically, each principal shares a secret with some trusted machine, called an authentication server. By proving possession of this secret, a principal can establish trust in its identity. The use of passwords in a multi-user environment is an example of this.

The shared secret in an authentication system is typically used as an encryption key. The encryption scheme has the property that a user cannot generate or decrypt encrypted data without possession of the key. Thus, a principal proves it is in possession of a key by encrypting with it.

Authentication in a large, distributed system is challenging because principals communicate over a network that is vulnerable to many attacks. A passive intruder can eavesdrop on a line and obtain sensitive information. Of graver consequence, is an active intruder who can modify message traffic by blocking the transmission of packets and inserting his own packets at will. Such an intruder can impersonate any principal in the system and possibly intercept his rights and privileges.

Encryption can thwart the attacks of an active intruder. Encrypted data has the property that any modification to some part of the data causes the decryption to fail. Thus, without knowledge of the key, an active, malicious intruder's ability is limited to blocking data from reaching its destination.

In authentication systems, we assume that each principal shares a secret key with an authentication server. This key is established by some secure, off-line method. Two principals can communicate securely by sending encrypted messages to the authentication server, who can re-encrypt and forward them to the intended recipient. However, issues of scale make this impractical.

Rather, when two principals wish to communicate, they establish a secret key known only to them. This secret key serves as a secure communication channel between the two principals because an active intruder who doesn't know the key cannot successfully interfere with the communication[1]. However, establishing such a key, called a session key, is a nontrivial problem.

The problem of establishing secure session keys between pairs of principals in a distributed authentication system led to a great deal of research. This research focuses on the development of protocols, and is accompanied by a greater and more interesting problem, the analysis of authentication protocols.

The Needham and Schroeder authentication protocol [40] revolutionized security in distributed systems. Adaptations of this protocol, such as Kerberos [54] and the Andrew File System [25] have become universal. However, it was not long before a flaw was found in this protocol [15]. Needham and Schroeder then published a revised version of their protocol [41].

The existence of a subtle flaw in a previously trusted protocol stressed the need for formal methods for analyzing authentication protocols. In fact, many authors praise the merits of their analysis techniques with their ability to discover the flaw in the Needham and Schroeder protocol [7, 10, 20, 37, 53, 63].

A few specification techniques for authentication protocols have been published [35, 60, 64, 67], and several formal analysis techniques have been proposed. In particular, the use of predicate logic for the analysis of protocols was proposed by Burrows *et al.*[2] [7], and many extensions have since been published [9, 10, 18, 20, 52, 53].

Others have been critical of the BAN logic [42, 57], and have proposed their own logics [30, 33, 35, 36, 37, 39, 55, 57, 63, 67]. This paper explores these logics and discusses the tradeoffs among them.

2 Terminology

This section describes some of the terminology used in the rest of the paper. Because many researchers define their own terms and use different notations, we have standardized on the

[1] It is assumed that systems will always be vulnerable to message blocking because in the simple case an intruder can cut the physical wire connecting two machines.

[2] This logic is referred to as BAN logic, after the authors Burrows, Abadi, and Needham.

following definitions.

Threat model refers to the assumed characteristics of the security environment. It includes the assumptions made about the principals involved and the possible interference of malicious agents. In this paper, the threat model includes an active intruder who can delete, modify, and create message traffic at will. We also assume strong encryption.

Encryption is the science or art of generating a cipher text from a clear text, making the clear text unrecognizable. In security systems encryption involves the use of a secret key and a known algorithm.

Decryption is the science or art of generating a clear text from a cipher text. In security systems decryption involves knowledge of a secret key and a known algorithm.

Cryptanalysis is the science or art of breaking a cryptographic code without knowledge of the key. The methods used take into consideration letter frequency, and any information about the context available. This type of analysis is very advanced and can defeat all but the best encryption techniques.

Strong encryption is an encryption method that is assumed to be computationally unbreakable. Also, it is not vulnerable to any form of cryptanalysis.

Z is a common notation to represent the intruder. (It is also common to see X and C.)

Key management protocol is used interchangeably with the terms *authentication protocols* and *cryptographic protocols*. It is a set of rules defining the messages passed in an encryption system to distribute secret keys.

Nonce is an identifier, usually a large random number, that is used only once. The main purpose of a nonce is to link two messages together so that a response can be recognized as fresh. A nonce is usually represented as N_a or N_b, etc.

Doxastic logic is based on belief. The reasoning system uses rules about how belief is propagated to establish new beliefs.

Epistemic logic is based on knowledge. The reasoning is similar to reasoning in a doxastic logic, but these logics are used to reason about knowledge instead of belief.

$\{\boldsymbol{data}\}_k$ represents *data* encrypted under secret key, k.

Session key is a secret encryption key established between two principals for communication purposes. As the name implies, this key is intended for one session only. Sometimes this session is only one protocol run; often it lasts for the lifetime of a ticket or token.

Symmetric keys are used for private key systems. In such systems, the same key is used for encryption and decryption. For example, $\{data\}_k$ can be decrypted with k.

Asymmetric (public) keys are pairs of keys that are inverses of each other. One key is kept private, and is known only to the principal who possesses it. The other key is public, and is made widely available. Data encrypted with the private key can be decrypted with the public key; similarly, data encrypted with the public key can be decrypted with the private key.

Symmetric protocol exists if two principals play the same role in the protocol. Thus a protocol in which a principal speaks with the authentication server is not symmetric, whereas a protocol, in which two users at the same trust level share data usually is symmetric.

3 Needham and Schroeder

We now turn to one of the most famous and landmark protocols to begin our discussion of protocol analysis.

The Needham and Schroeder protocol [40] distributes a secret session key between two principals in a network. The threat model of the Needham and Schroeder protocol assumes

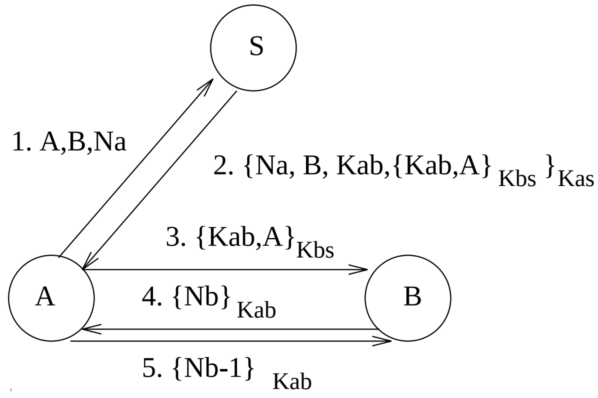

Figure 1: **The Needham and Schroeder Protocol**

that each principal shares a secret key with an authentication server and that an intruder can read and modify anything that passes on the network. In addition, the model assumes that intruders can block any message from reaching its destination and insert malicious messages of their own.

The participants in this protocol are the three principals, A, B, and S, where S is the authentication server, and A is a principal who wishes to initiate a secure session with principal B. Thus, as pointed out by Sidhu [50], this protocol is not symmetric. We represent a protocol step as

$A \rightarrow B : Message$

to indicate that A sends $Message$ to B. Thus, the Needham and Schroeder protocol can be specified as follows:

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
3. $A \rightarrow B : \{K_{ab}, A\}_{K_{bs}}$
4. $B \rightarrow A : \{N_b\}_{K_{ab}}$
5. $A \rightarrow B : \{N_b - 1\}_{K_{ab}}$

This protocol is represented graphically in Figure 1. Each node represents a principal, and the transitions represent the messages being sent. The transitions are numbered in the order of the messages. K_{ab} represents the secret key shared by A and B, etc.

In message 1, A sends a request to the server (S) indicating that it wishes to communicate with B. The nonce N_a is included to link future messages to this request. This message is sent in the clear because it includes no security related information.

In message 2, the server S responds with a session key, K_{ab}. A copy of the key is also encrypted under B's secret key. In addition, N_a is included as a guarantee that this message is not a replay of a previous response. Each principal is also told which principal will be on the other end of the secure channel. This can be seen by the inclusion of A in $\{K_{ab}, A\}_{K_{bs}}$.

In message 3, A forwards $\{K_{ab}, A\}_{K_{bs}}$ to B, who can decrypt it and recover K_{ab}. B then issues message 4 as a challenge to A to make sure that A possesses K_{ab}. In message 5, A proves possession of the session key. At the end of the protocol, it would seem that A and B would be in possession of K_{ab},[3] and that no intruder could possibly know the secret session key. Thus, this protocol appears to allow A and B to establish a secure channel.

3.1 A Weakness in the Protocol

Denning and Sacco [15] were the first to discover a major weakness in the Needham and Schroeder protocol.

It is assumed that a session key is meant to be used only once and then discarded. Now, if we assume an intruder, Z, has recorded a previous run of the Needham and Schroeder protocol, then an attack is possible if the old session key is compromised.

To illustrate, suppose that an old session key, CK, has been compromised. If Z recorded the protocol run where CK was established, then Z can replay the message:

$Z \rightarrow B : \{CK, A\}_{K_{bs}}$

Thinking A has initiated a new conversation, B requests a handshake from A:

$B \rightarrow A : \{N_b\}_{CK}$

Z intercepts the message, decrypts it with CK, and impersonates A's response:

[3] We ignore the fact that S also has K_{ab} because it is assumed to be a trusted server that would not abuse the key.

$Z \rightarrow B : \{N_b - 1\}_{CK}$

Thereafter, Z can send bogus messages to B that appear to be from A. B will have no way of knowing that it is not communicating with A.

3.2 Handling the Weakness

Denning and Sacco suggest that by adding timestamps to messages 2 and 3, the problem can be solved. Thus, these two steps become:

$S \rightarrow A : \{T, N_a, B, K_{ab}, \{K_{ab}, A, T\}_{K_{bs}}\}_{K_{as}}$

$A \rightarrow B : \{K_{ab}, A, T\}_{K_{bs}}$

where T is a timestamp. Thus, a replay of message 3 would be recognized as old and would be ignored.

In a follow-up paper Needham and Schroeder propose a solution that is based on the use of nonces[41]. They observe that one of the communicating parties will require proof of the timeliness of a future message. It is always this party that should generate the nonce identifier.

This is achieved as follows. Before the protocol takes place,

$A \rightarrow B : A$

$B \rightarrow A : \{A, J\}_{K_{bs}}$, where J is a nonce identifier that will be kept by B.

Now, J can be included in the authenticator sent to A to be forwarded to B. Thus, B will be assured that the session key is fresh and not a replay.

The vulnerability of the Needham and Schroeder protocols comes from the fact that each session key is meant for exactly one session. If an intruder can compromise an old session key, he can force its use in another session. Both Denning and Sacco's solution and the revised Needham and Schroeder protocols solve this problem by requiring that the forwarded message from A to B establish a new session.

This section deals with symmetric secret keys. The arguments are similar for public key systems, and we do not repeat them here.

3.3 Discussion

We have shown how a weakness discovered in a published protocol can be fixed, but we have not proved that the resulting protocol is secure. Furthermore, we have not shown that a mechanical technique could discover this weakness. In the remainder of this paper we will discuss how formal methods have been applied to the analysis of authentication protocols.

4 Approaches to Analysis

Meadows [36] defines four approaches that have been taken in the analysis of cryptographic protocols:

Type I– To model and verify the protocol using specification languages and verification tools not specifically developed for the analysis of cryptographic protocols.

Type II– To develop expert systems that a protocol designer can use to develop and investigate different scenarios.

Type III– To model the requirements of a protocol family using logics developed for the analysis of knowledge and belief.

Type IV– To develop a formal model based on the algebraic term-rewriting properties of cryptographic systems.

The Type I approach is the least popular, while the Type III approach is the most common. These approaches share a few properties. In all cases, the methods are independent of the underlying cryptographic mechanisms.[4] In addition, we typically assume a set of principals and a trusted authentication server. The principals are not trusted, and may consist of a privileged intruder who can add, delete, or modify messages on the network at will.

The next four sections describe each of the four types of authentication protocol analysis. Table 1 shows the focus of current research. The entries in the table refer to the bibliography reference numbers.

[4] For a good discussion of failures in a cryptosystem due to the underlying encryption mechanisms see Moore [38].

First Author	Protocol Specification	Protocol Analysis Type I	Type II	Type III	Type IV
Abadi				[1]	
Bieber				[2]	
Blumer	[3]		[3]		
Britton		[4]			
Burrows				[7] [8]	
Calvelli				[9]	
Campbell				[10]	
Dolev					[16]
Gaarder				[18]	
Gong				[19] [20]	
Gray				[26]	
Kailar				[27]	
Kasami					[28]
Kemmerer	[29]	[29]	[29]		
Longley			[30]		
Lu					[31]
Mao				[32]	
Meadows	[35]		[35]		[33] [34] [35] [36]
Millen			[37]		
Moser				[39]	
Nessett				[42]	
Rangan				[44]	
Sidhu			[50]		
Snekkenes				[51] [52] [53]	
Syverson	[60]		[60]	[56] [57] [58] [59] [61]	[55] [60]
Varadharajan	[62] [63] [64]	[62] [63] [64]			
Woo	[67]				[67]

Table 1: The Focus of Research in the Specification and Analysis of Authentication Protocols by Category. Entries in the table correspond to bibliography reference numbers. The four types under protocol analysis are as described by Meadows [36].

5 Type I Approach

The Type I approach to the analysis of cryptographic protocols is to model and verify protocols using specification languages and verification tools not specifically developed for the analysis of such protocols. The main idea is to treat a cryptographic protocol as any other program and attempt to prove its correctness. A criticism of this approach is that it proves correctness and not necessarily security [50].

The first step in this approach is to specify the cryptographic protocol in a way that the techniques being used can be applied. Sidhu [50] suggests a specification technique that involves representing a protocol as a directed graph. Varadharajan [63] also adopts this method. However, in a more recent publication [64], he uses LOTOS (Language of Temporal Ordering Specification) for specifying authentication protocols.

The work by Kemmerer [29] fits into several of the types of approaches, as shown in Table 1. The author describes an example system with a special cryptographic facility. The Type I approach can be seen in his attempt to use machine-aided verification techniques. The properties that the protocol should preserve are expressed as state invariants, and the theorems that must be proved to guarantee that the cryptographic facility satisfies the invariants are automatically generated by the verification system.

It should be noted that although much effort was concentrated on the Type I approach early on, most work in this area has been redirected as the logics of the Type III approach have gained popularity.

5.1 Using a Formal Verification System

Kemmerer [29] describes two goals in using formal methods for the analysis of encryption protocols. The first is to verify formally that an encryption protocol satisfies its stated security requirements, and the second is to discover weaknesses in its specification. His formal model uses a state machine approach where a system is viewed as being in various states, which are differentiated from one another by the values of state variables. The values of the variables can be changed only via well-defined state transitions.

Kemmerer uses an extension of first-order predicate calculus, a formal specification language called Ina Jo [48]. This nonprocedural assertion language was not developed specifically for use with security protocols, and thus this work fits into the Type I analysis approach.

Ina Jo uses the following symbols for logical operations:

& logical AND

$\rightarrow$ logical implication

In addition, there is a conditional form,

(if A then B else C)

where A is a predicate and B and C are well-formed terms. The notation for set operations is:

$\in$ is a member of

$\cup$ set union

{a,b,...,c} set consisting of elements a,b,...,and c

{set description} set described by set description

The language also contains the following quantifier notation:

$\forall$ for all

$\exists$ there exists

There are also two special Ina Jo symbols:

N'' to indicate the new value of a variable (e.g., $N''v1$ is the new value of variable $v1$)

T'' which defines a subtype of a given type, T

Kemmerer [29] describes an example system, and then gives an Ina Jo specification of the system. In this system, n terminals are connected to a central host. Each terminal contains a cryptographic facility that holds a permanent terminal key. The host stores two tables of keys. The first table is a list of the session keys being used in the system, and the second table contains the terminal keys. As such, the host acts as an authentication server.

In this system, the host is connected to a tamper-proof cryptographic facility that holds master keys for decrypting the information in the two tables. This system is used for pedagogical purposes and has not actually been implemented. The system architecture is shown in Figure 2[5]. Ina Jo *constants* and *variables* are described, along with *transforms*. An example of a constant in this example system is:

Terminal_key(Terminal_num):Key

because each terminal has a constant terminal key. However, as session keys vary from session to session, an example variable in Ina Jo is:

Session_Key(Terminal_num):Key

An example of a transform in Ina Jo is *Generate_Session_Key*. These are used to change state in the analysis.

An Ina Jo *axiom* is an expression of a property that is assumed. For example, to express that encryption and decryption are commutative, we would use the following Ina Jo axiom:

[5] This figure is based on the figure by Kemmerer [29].

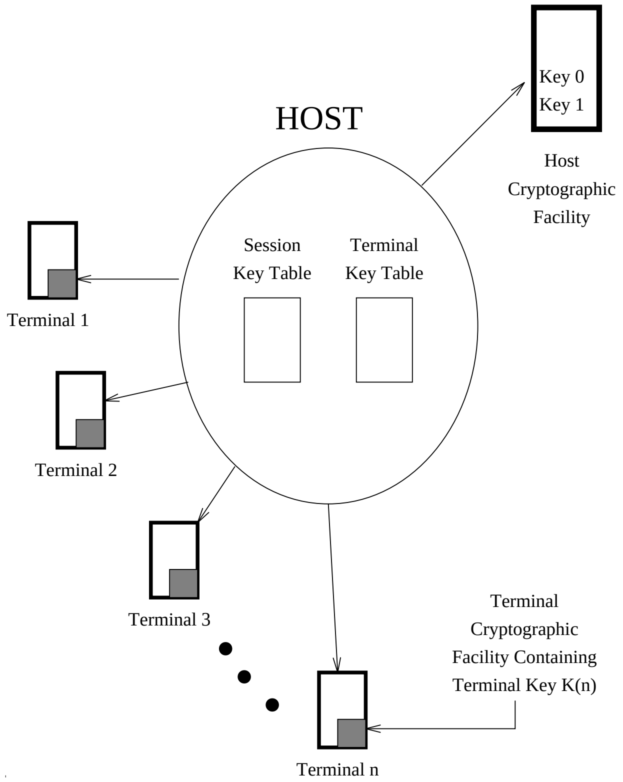

Figure 2: **System Architecture for Kemmerer's Sample System.**

AXIOM $\forall t$:TEXT,
$k1, k2$:Key (Encrypt($k1$, Decrypt($k2, t$)) = Decrypt($k2$,Encrypt($k1, t$))).

Other such axioms are given in the full specification found in the appendix of Kemmerer's paper [29].

Finally, Ina Jo *criteria* clauses are used to specify the critical requirements that the system is to satisfy in all states. For example, the criterion that no key available to the intruder can be used for encryption can be specified as:

CRITERION $\forall k$:Key ($k \in$ Intruder_Info $\rightarrow$ $k \notin$ Keys_Used).

Once the specification is complete, Ina Jo generates theorems that can be used to verify if the critical requirements (criterion) are satisfied. Kemmerer points out that "an advantage of expressing the system using formal notation and attempting to prove properties about the specification is that, if the generated theorems cannot be proved, the failed proofs often point to weaknesses in the system or to an incompleteness in the specification."

Kemmerer uncovers a weakness in his sample system using the formal specification. However, the value of this method is limited because proving the criterion of an Ina Jo specification does not necessarily guarantee that a protocol is secure. In addition, to specify requirements that secure a system from active attacks, the designer first needs to know the potential attacks, obviating any need for formal methods to discover them.

5.2 Using LOTOS for Protocol Specification

Varadharajan [64] proposes the use of LOTOS to analyze authentication protocols. He gives as examples the specification of two protocols that have been adopted as standards: the ISO/DP 9798 and CCITT X.509. However, no results are given. The paper concludes by stating that LOTOS tools are not yet adequate and are currently being developed.

The paper gives a very strong recommendation for the use of LOTOS. The goals of such a Formal Description Technique (FDT) are outlined as follows:

expressive power: ability to express a wide range of properties required for the description of services and protocols.

well-defined: syntax and semantics enabling mechanical manipulation, and validation.

well-structured: increasing understandability and maintainability of specifications.

abstraction: allowing representation of architectural aspects at a sufficiently high level of abstraction, where implementation details are not specified.

LOTOS has been developed for systems related to the Open Systems Interconnection (OSI), and is based on a process algebra that does not use a temporal logic, despite what the name might imply.

A system in LOTOS is modeled as a collection of processes in which the order of events is specified. As such, it can be used to model the messages sent in an authentication protocol. However, to date no concrete results have been reported using this method.

Methods of Type I will have to demonstrate some success before they become popular. The next section describes another attempt to use tools not originally intended to analyze authentication protocols.

5.3 Specifying a Protocol as a Finite State Machine

Sidhu [50] and Varadharajan [63] describe how to specify a protocol using state diagrams. A directed graph is used for each principal. First, an initial state is specified. Then, an arc is drawn to another state for each message that can be sent or received at that point. We will demonstrate this with an example.

The Needham and Schroeder [40] protocol is reproduced below for reference.

1. $A \rightarrow S : A, B, N_a$
2. $S \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
3. $A \rightarrow B : \{K_{ab}, A\}_{K_{bs}}$
4. $B \rightarrow A : \{N_b\}_{K_{ab}}$
5. $A \rightarrow B : \{N_b - 1\}_{K_{ab}}$

We use the following notation:

P^{-1} event – principal P transmits message 1

P^{+1} event – principal P receives message 1

Varadharajan [63] gives a state diagram for each entity, A, B, and S. The attempt is to capture the behavior of each principal in the protocol. However, his example is highly complex and counterintuitive. We prefer to represent the protocol as a cross product of the state diagrams for each individual principal (Figure 3). The nondeterministic finite state machine is constructed from the individual machines for A and B. The individual machine for a principal is composed of a sequence of states with arcs representing the transmission or reception of a message. A state is labeled P^{-n} to indicate that principal, P has transmitted message number n and P^{+n} if P receives message n.

If the final accepting state is reached, then we have a legal run of the protocol initiated by either A or B. If an individual principal's machine consists of x states, then the cross product machine with another principal in the protocol has $x^2 + 1$ states including the final accepting states. All other states represent illegal runs of the protocol.

As we describe each state in our protocol specification, notice that it is assumed that A and B play the same role in the protocol. This assumption is controversial. Varadharajan [63] states that "A and B have symmetric roles." However, Sidhu [50] states that "The authentication protocols of Needham and Schroeder are not symmetric between a sender and a receiver and assume a particular time ordering of events."

The state representations presented by Sidhu differ slightly from those of Varadharajan. Remaining impartial, we present our own state diagram construction, and refer the curious reader to Sidhu's and Varadharajan's papers [50, 63] for their representations.

The next section discusses how these finite state machines that are used to specify protocols can also be used in their analysis.

5.4 The Use of Finite State Machines for Protocol Analysis

The state machines described above can be used to analyze authentication protocols by employing a technique known as the reachability analysis technique [66].

To use this technique, for each transition, the global state of the system is expressed using the states of the entities and the states of the communication channels between them. Each global state is then analyzed and properties are determined, such as deadlock and correctness. If an entity is not able to receive a message that it is supposed to receive in a given state, then there is a problem with the protocol. For an example of such an analysis, see Varadharajan [63].

Reachability analysis techniques are effective in determining whether or not a protocol is correct with respect to its specifications, the purpose for which they were invented. However, they do not guarantee security from an active intruder. The weakness of Type I analysis techniques is that in applying methods that were

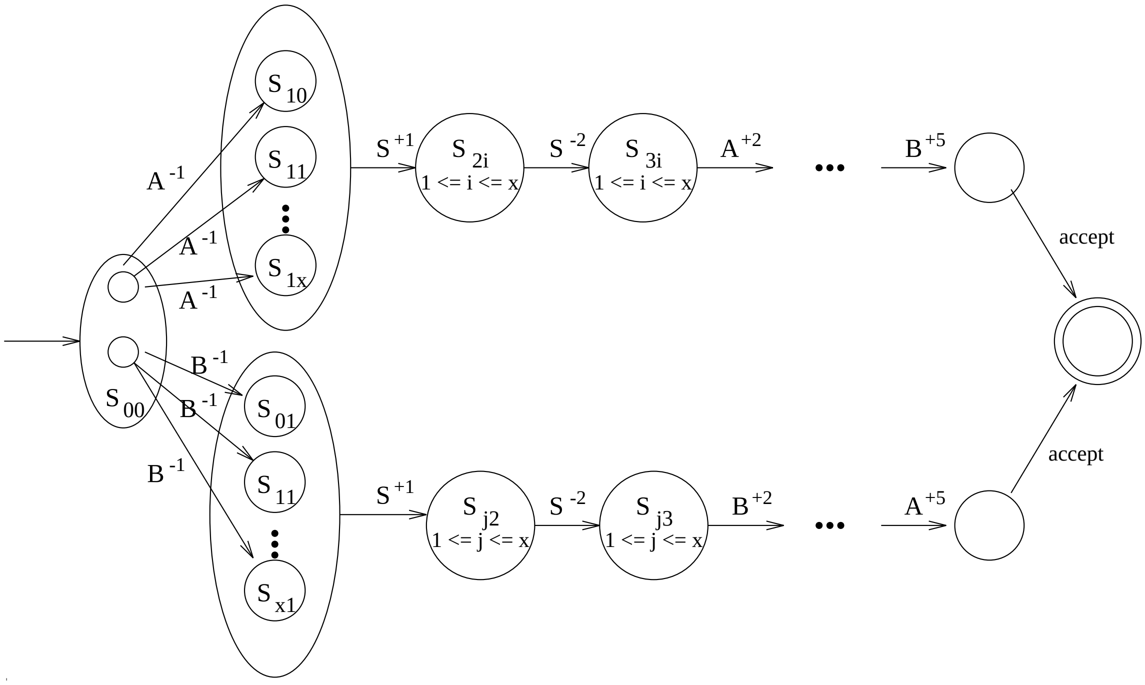

Figure 3: **Nondeterministic Finite State Machine for Principals A and B Intiating the Needham and Schroeder Protocol.** The arc P^{-n} means that principal P transmits message number n. P^{+n} means that P receives message n. This machine is constructed by taking the cross product of the individual machines for A and B initiating the portocol. If a state of A's machine is S_i, and B's is labeled S_j, then the corresponding state in the cross product machine is S_{ij}. The number of legal states in each of A's and B's machines is x, and the cross product contains $x^2 + 1$ legal states including the accepting final state. All other states are illegal, and stand for illegal runs of the protocol.

not intended specifically for security analysis, subtle pitfalls that are peculiar to the security domain, such as the effect of message replay, are not considered.

6 Type II Approach

The Type II approach to the analysis of cryptographic protocols is to develop expert systems that a protocol designer can use to develop and investigate different scenarios. These systems begin with an undesirable state and attempt to discover if this state is reachable from an initial state.

Although this approach may better identify flaws than Type I approaches, it does not guarantee the security of an authentication protocol, nor does it provide an automated technique for developing attacks on a protocol. In other words, the Type II approaches can discover whether a given protocol contains a given flaw, but are unlikely to discover unknown types of flaws in protocols.

Longley and Rigby [30] summarize the value of expert systems in the analysis of key management schemes. The expert systems provide:

- a new perspective on an authentication system;
- a technique of building models capable of continuous refinement;
- a method of interaction with the model, which provides a greater insight into the operation of the system;
- a model that responds to *what if* questions; and

- a method of testing the effects of proposed system modifications.

Thus, expert systems can be used in conjunction with other analysis techniques such as those of Type III and IV for the purposes mentioned above, but they will never replace those techniques.

The NRL protocol analyzer [60] might be viewed as a Type II approach. However, because it is based on the Dolev and Yao model [16], in which an intruder produces words in a term-rewriting system, we will consider it a Type IV approach.

6.1 The Interrogator

The Interrogator, by Millen *et al.* [37] is a noteworthy effort to apply expert systems to the analysis of security protocols. The input to the system is a protocol specification and a target data item. The output is a message history showing how the penetrator could have obtained this data item.

In the Interrogator, a protocol is viewed as a collection of communicating processes, one for each principal. Each process has a set of possible states, and the transmission of a message can cause a state transition in a process. Each process maintains its own state, and when applicable, sends messages to other processes causing them to change state. The system is based on the finite state machine approach [24].

The Interrogator generates a large number of paths through a protocol, ending in a specified insecure state. If any of these paths start with an initial state, then a vulnerability has been discovered. Thus, an important issue in using the Interrogator is the specification of the final state.

In the Interrogator, the penetrator is expressed as a relation:

`p_knows`(x, H, q)

where x is the data item learned by the penetrator, H is the message history that lead to this discovery, and q is a state of the network reachable from the initial state. The meaning of `p_knows` is as follows:

`p_knows`(x, H, q) **iff**

x is known initially

or $(H = H'\text{sent}(m)$ **and** $\text{sent}(m) : q' \rightarrow q$ **and** $H' : q_0 \rightarrow q'$ **and** `p_gets`$(x, m, H', q'))$

or $(H = H'e$ **and** $e : q' \rightarrow q$ **and** `p_knows`$(x, H', q'))$

or $(H : q_0 \rightarrow q'$ **and** `p_modifies`(q', q, H) **and** `p_knows`$(x, H, q'))$

Similarly, `p_gets` is defined as follows:

`p_gets`(x, m, H, q) **iff**

x is a field of m

or $(\{m'\}_k$ is a field of m **and** `p_knows`(k, H, q) **and** `p_gets`$(x, m', H, q))$

The definition of `p_knows` describes the three ways a penetrator may learn x with message history, H, in state q. The penetrator may learn it from the last message read; may have already known it in the previous network state, q'; or may learn it using `p_modifies` described below.

The definition of `p_gets` states that a penetrator can read any message, but if some part of the message is encrypted, then it can only be extracted if the key encrypting that field is known. The statement `p_modifies`(q', q, H), describes how a penetrator who modifies the network buffer can learn x. Millen *et al.* state that

> "`p_modifies`(q', q, H) is characterized by saying that if m is a new message in the network buffer of the new state q, the penetrator knows each field of m in the prior state q' reached by history H." [37]

This means that if the penetrator knows x in state q', reachable with message history H, and the penetrator changes the message buffer such that state q is reached instead with message history H, the penetrator still knows x.

Millen *et al.* claim that the Interrogator was able to rediscover the flaw in the Needham and Schroeder protocol. However, the Interrogator was first provided with information to the effect that the penetrator knows an old connection key. This information could be supplied because the programmers of the Interrogator were familiar with the weakness in the Needham and Schroeder protocol.

Systems such as the Interrogator can be useful for providing message histories for *known* attacks, but it remains to be seen whether such methods will discover new attacks on protocols previously believed to be secure. No such result has been reported.

6.2 A Rule-Based System

Longley and Rigby [30] describe a rule-based system used to test the vulnerability of a key management scheme to specified attacks. The results of applying this system to the IBM key management scheme described by Davies and Price [14] were consistent with the known characteristics of that scheme.

The expert system uses an exhaustive search to determine if a given attack is successful. When the system halts, then the history of rule firings can give an attack strategy. Until then, nothing can be said about the given attack. In fact, in some cases, the search space is infinite and the system does not even halt.

Longley *et al.* use a rule-based system, OPS5 [5]. This system uses rules to transform goals into sub-goals, and this process is continually refined until a concrete attack strategy is reached.

At best, this system can be used as a model of threat analysis. It does not perform the function of analysis in terms of demonstrating the security of an authentication protocol. Rather, it can *sometimes* determine how a given attack might be successful against a protocol.

6.3 Discussion

The Type II approaches to protocol analysis serve a limited function. They are most useful for analyzing known weaknesses in protocols, and generating message lists to exploit those weaknesses.

The systems developed under this approach are usually inefficient, often resorting to exhaustive search. In addition, the results are often inconclusive, and the systems may not even halt.

Their limitations are due to the lack of expressiveness of the types of rules found in expert systems. For this reason, the majority of research into the analysis of authentication protocols falls into the Type III category, discussed next.

7 Type III Approach

The Type III approach to protocol analysis uses formal logic models developed for the analysis of knowledge and belief. Burrows *et al.*'s landmark BAN logic [7] initiated intense research using this approach. Since then, BAN has been extended [9, 10, 18, 20, 32, 52], and criticized [32, 42, 52, 57].

This section discusses other contributions to the Type III approach including the logic of Bieber [2] and its extension by Snekkenes [53]; the axiomization of trust and belief by Rangan [44]; the logic of Syverson [55]; the logic of Kailar *et al.* [27]; and the logic of Moser [39].

In addition to these logics, some work has concentrated on the semantics of logics for authentication protocols [1, 61]. We discuss the semantics introduced here and why it is important to define the semantics of a logic with great care.

In the following sections we present these logics, discuss, compare, and evaluate their relative merits.

7.1 An Axiomization of Belief

Much of the work in the Type III approach is based on a formal axiomization of belief and trust. Shoham and Moses [49] describe the relationship between knowledge and belief, and note a close connection between belief and nonmonotonic reasoning.

Syverson [61] shows that belief and knowledge are equally adequate for protocol analysis on the logical level. Logics based on knowledge are termed epistemic, while doxastic logics refer to those based on belief. The main difference in reasoning with these two logics is that all epistemic logics have an axiom that states that if a principal knows X, then X. No doxastic logics have such an axiom. However, Syverson shows that this axiom (termed axiom **T**) can easily be captured in doxastic logics.

Rangan [44] provides an axiom schema for belief that is frequently referenced in the literature. In his notation, the term $B_i p$ means that principal i believes p. The schema is as follows.

for all $i, i = 1, ..., m$:

A1 All substitution instances of propositional tautologies.

A2 $B_i p \wedge B_i(p \Rightarrow q) \Rightarrow B_i q$.

A3 $B_i p \Rightarrow B_i B_i p$ (introspection of positive belief).

A4 $\neg B_i p \Rightarrow B_i \neg B_i p$ (introspection of negative belief).

A5 $\neg B_i(\text{false})$ (process i does not believe a contradiction).

The following are the inference rules.

for all $i, i = 1, ..., m$:

R1 From p and $p \Rightarrow q$ infer q (modus ponens).

R2 From p infer $B_i p$ (generalization).

In his paper [44], Rangan defines the *transitivity, Euclidian,* and *serial* properties, and shows that **A3** corresponds to transitivity, given **R2**, **A4** corresponds to the euclidian property, and **A5** corresponds to the serial property.

These definitions of knowledge and belief are the foundation for the Type III approaches discussed below.

7.2 The BAN Logic

The BAN logic casts authentication protocols in formal terms to reason about the state of belief among principals in a system. The authors' goals were to be able to answer the following questions about a protocol:

- What does this protocol achieve?
- Does this protocol need more assumptions than another one?
- Does this protocol take any unnecessary steps, ones that could be left out without weakening it?
- Does this protocol encrypt a message that could be sent in the clear without weakening security?

The authors state that such issues as errors introduced by concrete implementations of a protocol, such as deadlocks, or inappropriate use of a cryptosystem (as described by Voydock and Kent [65]) are not considered; this system deals with authentication protocols on an abstract level only.

7.2.1 The basic constructs of the BAN logic

The only propositional connective is conjunction, which is denoted with a comma. Associativity and commutativity properties are taken for granted.

P *believes* X The principal, P, acts as though X is true.

P *sees* X Someone has sent a message containing X to P, who can read and repeat X (possibly after doing some decryption).

P *said* X At some time, the principal P sent a message that includes the statement X. It is not known how long ago the message was sent, or even if it was sent during the current run of the protocol. It is known that P believed X when he said it.

P *controls* X The principal P is an authority on X and should be trusted on this matter. This construct is used primarily when a principal has delegated authority over some statement.

#(X) The formula X is *fresh*. That is, X has not been sent in a message at any time before the current run of the protocol. This is defined to be true for *nonces*, that is, expressions generated for the purpose of being fresh.

$\mathbf{P} \overset{K}{\leftrightarrow} \mathbf{Q}$ P and Q may use the *shared key* K to communicate. The key K is good, in that it will never be discovered by any principal except P or Q, or a principal trusted by either P or Q.

$\overset{K}{\mapsto} P$ P has K as a *public key*. The matching *secret key*, K^{-1}, will never be discovered by any principal other than P or a principal trusted by P.

P$\overset{X}{\rightleftharpoons}$Q The formula X is a secret known only to P and Q, and those principals to whom they reaveal it. P and Q may use X to prove their identities to one another.

$\{$X$\}_K$ from P This represents the formula X encrypted under the key K by principal P. The from P part is often omitted, and it is assumed that each principal is able to recognize and ignore his own messages.

Logical postulates are formed from these basic constructs. A security protocol is idealized, according to rules defined by the authors, in terms of these postulates. Every protocol must be idealized before using the BAN logic; many examples follow.

7.2.2 The rules of inference of the BAN logic

Burrows *et al.* [7] provide rules of inference for reasoning about the belief in a protocol. These rules are applied to the initial assumptions to drive a proof or to answer questions about a protocol. One important rule, the *message meaning rule*, states how to derive belief from the origin of a message.

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

Remember that $\{X\}_K$ in this context stands for $\{X\}_K$ from R $\neq$ P. Then this formula can be intuitively explained as:

> IF P believes that Q and P share a secret key, K, and P sees X, encrypted under K, and P did not encrypt X under K, THEN P believes that Q once said X.

A similar postulate exists for public keys and shared secrets. Another important rule of inference for the BAN logic is the *nonce-verification rule*.

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

For the sake of simplicity, the authors of BAN state that X must be clear text, that is, it should not include any subformula of the form $\{Y\}_K$. An intuitive explanation of this rule is:

> IF P believes that X could have been uttered only recently and that Q once said X, THEN P believes that Q believes X.

This rule is important because many protocols rely on the use of nonces to avoid successful replay attacks. In fact, this is the only postulate that promotes from *said* to *believes*, and thus reflects in an abstract way, the practice of using challenges and responses for authentication. A result of applying this rule demonstrates that challenges often need not be encrypted, but responses must be.

The next rule, the *jurisdiction rule*, is often used for delegation.

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

This rule states:

> IF P believes that Q has jurisdiction over X, and P believes that Q believes X, THEN P believes X.

Burrows *et al.* [7] provide many other inference rules that can be used to combine beliefs.

7.2.3 The idealized protocol of the BAN logic

For a protocol to be analyzed using the BAN logic, it must first be converted to an idealized form. Typically, a step in a protocol is written as:

$$\text{P} \rightarrow \text{Q : message}$$

This means that P sends the message and that the principal Q receives it. This framework is often ambiguous, and does not lend itself to formal analysis. For example, when something is encrypted under a session key, it may not always be clear what parts of the message are fresh, or who exactly knows this key. Therefore, each step in a protocol is transformed into an idealized form. A message in the idealized protocol is a formula. Say we define the protocol step:

$$A \rightarrow B : \{A, K_{ab}\}_{K_{bs}}$$

In this step, A tells B, who knows the key, K_{bs}, that K_{ab} is a key to communicate with A. It is clear that A did not generate this message, because A does not know K_{bs}. In fact, the message must have come from the server S. This step is idealized as:

$$A \rightarrow B : \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$$

When this message is sent to B, we can deduce that the formula

$$\text{B sees } \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$$

holds, indicating that the receiving principal becomes aware of the message and can act upon it.

In the idealized form, parts of the message that do not contribute to the beliefs of the recipient are omitted. Thus, clear text parts of the message are not included, because they can be intercepted and read or forged by anyone. Idealized messages are of the form $\{X_1\}_{K_1}$,...,$\{X_n\}_{K_n}$, where each encrypted part is treated separately.

The authors of BAN logic "view the idealized protocols as clearer and more complete specifications than the traditional descriptions found in the literature, which we view merely as implementation-dependent encodings of the protocols" [7]. However, no clear transformation method is presented. The paper gives numerous examples of the transformation to an idealized protocol; after careful study, idealizing protocols becomes intuitive. However, Woo and Lam [67] criticize the idealization of protocols. "We find idealization undesirable because of the potentially large semantic gap that exists between the original protocol and the idealized version."

Nessett's criticism [42] raises similar concerns, as we shall see.

7.2.4 Protocol analysis with the BAN logic

The steps in protocol analysis with BAN logic as presented by its authors are:

1. The idealized protocol is derived from the original one.

2. Assumptions about the initial state are written.

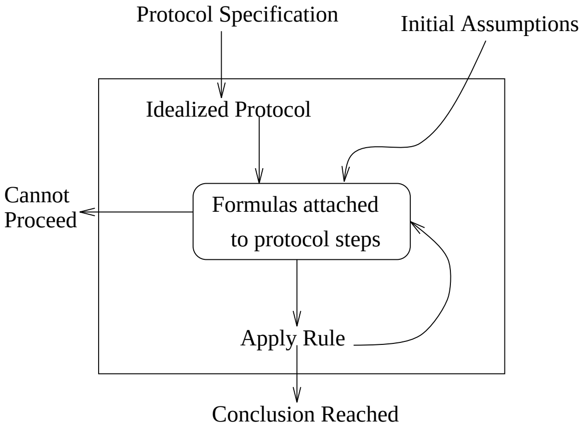

Figure 4: **Protocol Analysis with the BAN Logic:** The input to BAN is a protocol specification and the initial assumptions. At each step, formulas are attached to the protocol messages, and either a rule is applied, or the logic must halt. If possible, the desired conclusion is reached.

3. Logical formulas are attached to the statements of the protocol, as assertions about the state of the system after each statement.

4. The logical postulates are applied to the assumptions and the assertions to discover the beliefs held by the parties in the protocol.

More precisely, a protocol in the BAN logic is an ordered series of "send" statements, S_1,...,S_n, each of the form P $\rightarrow$ Q : X with P $\neq$ Q. An *annotation* for a protocol consists of a sequence of assertions inserted before the first statement and after each statement. The assertions are made by combining formulas of the forms P *believes* X and P *sees* X. The first assertion contains the assumptions, while the last assertion contains the conclusions. These are similar to simple formulas in Hoare logic [23]. They are written in the form:

$$\begin{gathered}\text{[assumptions]}\\ S_1 \text{ [assertion 1] } S_2 \ldots \text{ [assertion } n-1\text{] } S_n\\ \text{[conclusions]}\end{gathered}$$

Protocol analysis with the BAN logic is summarized in Figure 4. The protocols use no notion of time. Instead, time is divided into past and present depending on whether something was said in a previous or current run of the protocol.

The authors of BAN state that "More ambitious proofs may require finer temporal distinctions, reflected by constructs to reason about additional epochs, or even general-purpose temporal operators (see, for example, Halpern & Vardi 1986 [21])." In a recent paper, Syverson [58] introduces temporal axioms to the BAN logic and exposes protocol flaws using this extended logic.

7.2.5 The goals of authentication of the BAN logic

There is some debate as to what the goals of authentication are. Some argue that authentication is complete between A and B if there is a K such that:

$$\text{A believes A} \overset{K}{\leftrightarrow} \text{B}$$

$$\text{B believes A} \overset{K}{\leftrightarrow} \text{B}$$

Others believe that an authentication protocol should achieve:

$$\text{A believes B believes A} \overset{K}{\leftrightarrow} \text{B}$$

$$\text{B believes A believes A} \overset{K}{\leftrightarrow} \text{B}$$

The first set of goals is referred to as first-level belief, whereas the second set is termed second-level belief. According to Syverson [57], the level of belief needed varies for different applications and should be specified along with the protocol; the goals of BAN logic have often been misinterpreted.

Cheng and Gligor [12] claim the following conditions for the BAN logic must be satisfied at the end of a protocol run:

1. Both A and B believe K_{ab} is a secret key shared exclusively between A and B.

2. Both A and B believe that the other has the first-level belief. This is the second-level belief. If a party holds a second-level belief, then it believes that a secure channel has been established.

3. The causal relation between the first-level and second-level belief holds. That is the first level-belief must be established at some time before the second-level belief.

4. K_{ab} should be distributed exclusively to A and B; thus no parties other than A and B should have beliefs about K_{ab} ([12], p. 222).

Syverson [57] claims that these goals contradict the original goals set out by Burrows *et al.* In particular, using BAN logic, any principal who A and B trust can also be delegated the key, K_{ab}. Also, as Syverson points out, a second-level belief is not mandated by Burrows *et al.*, as Cheng and Gligor claim.

7.2.6 Nessett's criticism of the BAN logic

The BAN logic has been successful in finding flaws in some well known protocols, such as the Needham-Schroeder protocol, the Andrew secure RPC handshake, and the CCITT X.509 protocol. In addition, BAN has uncovered redundancy in the Needham-Schroeder conventional key protocol, the Otway-Rees protocol, Kerberos, the Yahalom protocol, the Andrew RPC handshake, and the CCITT X.509 protocol. As such, BAN logic can be called a success.

However, there are some problems with BAN logic. One problem is pointed out by Nessett [42], who demonstrates what he claims to be the hazards of devising systems of logic. He states that "a simple example shows that the BAN logic is capable of deducing characteristics about security protocols that by inspection are obviously false."

In Nessett's example, two principals, A and B communicate using public keys. The protocol is:

$$A \rightarrow B : \{N_a, K_{ab}\}_{K_a^{-1}}$$

$$B \rightarrow A : \{N_b\}_{K_{ab}}$$

The idealized form presented by Nessett is:

$$A \rightarrow B : \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_a^{-1}}$$

$$B \rightarrow A : \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}$$

A sends B a message, containing K_{ab}, the secret key between A and B, encrypted under A's private key. Thus, as the corresponding public key is well known, the key K_{ab} is no longer a secret. In the example, B then responds with a nonce identifier N_b, encrypted under the shared key, K_{ab}. According to the BAN model, this nonce is fresh and secret. However, as Nessett points out, it is obvious that N_a is readable and forgeable by anyone.

The problem with the BAN logic is that there is no way to represent what a principal does *not* know. All of the constructs and postulates deal with what a principal *does* believe, but there is no way to represent that a principal cannot know something. As Nessett states [42] "The essence of this flaw rests in the inability of the logic to analyze security protocols to assure that private information remains private."

Burrows *et al.* [8] defend their logic. They claim that the main difficulty in BAN logic as pointed out by Nessett is the assumption that A believes K_{ab} is a good shared key for A and B. "This assumption is clearly inconsistent with the message exchange, where A publishes K_{ab}. The inconsistency is not manifested by our formalism, but is not beyond the wit of man to notice."

Syverson [57] states that the confusion arises because "the BAN logic deals only with trust and not with security." Thus, he claims that Nessett's criticism is not valid because BAN does not claim to provide security, but rather, trust.

On the other hand, Snekkenes [52] attributes the Nessett flaw to the BAN logic being restricted to partial correctness. He defines a class of protocols called *terminating*, and shows that the Nessett protocol is a non-terminating one. "A statement or protocol step S terminates after finite time only if FALSE is not a derivable assertion succeeding S."

The criticism by Nessett has sparked a debate that has led to a clearer understanding of the role of knowledge and belief in the analysis of key management schemes. Much work has refered back to the research of Shoham and Moses [49] that specifically defines the relationship between belief and knowledge.

7.3 Extensions to the BAN Logic

The BAN logic was purposely designed to be open ended. That is, new constructs and postulates can be added to suit a particular application. It is not unusual to customize the BAN logic for an application to analyze a protocol to which the original BAN logic does not directly apply.

Some of these extensions have focussed on eliminating some of the assumptions in the original BAN logic. Others have been necessary for expanding the reasoning power of BAN. In the following sections we discuss some extensions to the logic.

7.3.1 The GNY logic

The Gong, Needham and Yahalom [20] extensions to the BAN logic are often referred to as the GNY logic. Gong *et al.* describe new constructs that eliminate some of the assumptions made by the original BAN logic. In particular, the GNY logic does not assume that redundancy exists in encrypted messages. Instead, they introduce the notion of *recognizability* to represent the fact that a principal expects certain formats in the messages it receives. Also, Gong *et al.* explicitly represent whether a principal generated a message itself.

The notion of *recognizability* is important. A principal participating in a protocol has expectations about the messages he will receive, and the analysis technique should take these into consideration. Thus, if a protocol step specifies that A will receive nonces, N_a and N_b, then the next two values received will be treated as nonces. Logical postulates are added to require that a principal's expectations according to the protocol are met. These rules are defined below.

One of the important contributions of the GNY logic is the recognition that belief and possession are different. In this extended logic, each principal maintains a *belief set* and a *possession set.* Along with the basic constructs of BAN, the following are included in the GNY logic:

P $\triangleleft$ X P is told formula X. P receives X, possibly after performing some computation such as decryption. A formula being told can be the message itself, as well as any computable content of that message.

P $\ni$ X P possesses, or is capable of possessing formula X. At a particular stage of a run, this includes all the formulae that P has been told, started the session with, or was able to compute for formulae he already possesses.

ϕ(x) The formula X is recognizable. If P *believes* ϕ(x), then P would recognize X if P had certain expectations about the nature of X.

P $\propto$ X P is eligible to send formula X. A principal is only eligible to send something that he possesses or can construct. P is eligible to send formula X. A principal is only eligible to send something that he possesses or can construct.

A formula in GNY may also be regarded as a *not-originated-here* formula, meaning that it was not previously generated by a principal in the current run. This is represented by adding an aserisk (*) to the formula; and the paper [20] describes a mechanical process by which this is achieved.

The following postulates are defined:

$$\frac{\mathrm{P} \triangleleft \mathrm{X}}{P \ni X}$$

which states that principals possess what they are told, and

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

which states that if a principal possesses X and Y, he also possesses the concatenation of X and Y, and any computable function F of X and Y. Similarly for recognizability,

$$\frac{\mathrm{P}\ \textit{believes}\ \phi(\mathrm{X})}{P\ \textit{believes}\ \phi(X,Y), P\ \textit{believes}\ \phi(F(X))}$$

states that if a principal believes that X is recognizable, then that principal believes that the concatenation of X with anything is recognizable, and that the application of some computable function to X is recognizable.

An important postulate in GNY states that if

$$\frac{C1}{C2}$$

is a postulate, then for any principal, P, so is

$$\frac{P\ \textit{believes}\ C1}{P\ \textit{believes}\ C2}$$

This is called the rationality rule and allows principals to reason about the state of other principals.

Gong *et al.* define preconditions that can be attached to rules to achieve different levels of belief. "Since we do not require the universal assumption that all principals are honest and competent, we should reason about beliefs held by others based on trust of different levels." Precondition statements are attached to formulas, and GNY provides trust and jurisdiction postulates for reasoning with preconditions.

The GNY logic is used to uncover the weakness in the Needham and Schroeder protocol. Then, the enhanced Needham and Schroeder protocol is analyzed and a second-level belief is attained. This logic is an improvement over the BAN logic in that it separates the content from the meaning of a message. Thus, the results of an analysis will be determined by the level of trust placed between principals. The original BAN logic did not accommodate for different levels of trust. Thus, the GNY logic increases the classes of protocols that can be analyzed.

In a later paper, Gong describes enhancements to the GNY logic to handle infeasible protocol specifications [19]. The problem is that a specification that could not possibly represent a real world situation can still be verified to be correct in the BAN and GNY logics. An example of such an infeasible specification is a protocol in which P sends R's password to Q. If R's password is represented by X, then this protocol step is:

$$P \rightarrow Q:\ \{X\}_{K_{PQ}}$$

As P and Q are not supposed to know R's password, this protocol is not feasible, and yet the GNY logic could not detect this.

Another type of infeasible specification that GNY and BAN cannot detect can lead to beliefs that do not preserve a causal relation. Say we have the protocol step:

$$P \rightarrow Q:\ \{P\ \textit{believes}\ P \stackrel{S}{\leftrightarrow} Q\}_{K_{PQ}}$$

This will cause Q *believes* P *believes* $P \stackrel{S}{\leftrightarrow} Q$, however, if it is not the case that the statement P *believes* $P \stackrel{S}{\leftrightarrow} Q$ already existed, then the causal relation between beliefs is not preserved. Without guarantees of causality, part of a causal chain may be broken, and then the path may not be trusted (e.g. [27]). The causal chain is broken any time a principal, P sends a message that contains a belief, and P does not hold that belief.

The BAN logic assumes that principals believe what they say, and so the infeasible specification due to causal relations is not an issue. Gong introduces the notion of *eligibility* to the GNY logic. Thus, if $P \propto X$, then P is eligible to send formula X. Thus, new postulates are added to the logic:

$$\frac{P \rightarrow Q : \; X, P \propto X}{Q \triangleleft X}$$

This rule states that if P sends X to Q, and P is eligible to send X, then Q receives X. Similarly,

$$\frac{P \ni X}{P \propto X}$$

This rule says that if P possesses X, then P is eligible to send X. Other rules are included to describe when a principal is eligible to encrypt with a key, K, or to perform a hash function.

Thus, using the extensions to the GNY logic, we can reason about protocols whose specification fall into two categories of infeasibility. Although Gong's work is useful, it appears that a more formal technique is needed to discover infeasible protocols in the general case.

7.3.2 The Mao and Boyd logic

Mao and Boyd [32] describe four weaknesses in the BAN logic and propose a new logic, based on BAN, which offers several improvements. Mao *et al.* also present a fault in a version of the Otway-Rees protocol [43] that the authors of the BAN logic proved to be correct [7]. Finally, a revised version of Otway-Rees is presented by Mao *et al.* and proved correct with the new logic.

Mao and Boyd discuss the following defect of the BAN logic:

1. Protocol Idealization
2. Belief
3. Protocol Assumptions
4. Confidentiality

The authors discuss these defects and give examples.

Protocol idealization suffers from too much flexibility. New terms and constructs can be freely chosen from an infinite alphabet provided by the original BAN logic. Also, as Mao *et al.* point out, "There seems to be no well-understood semantic rule to govern this job of idealization." [32]

The flaw in protocol idealization can be seen clearly if we view a protocol specification as a procedure declaration. Once formal parameters are substituted with "real values," the behavior is unpredictable. The authors make this analogy and and use it to demonstrate the flaw in the Otway-Rees protocol. The GNY logic [20] is also criticized for using the same idealization rules, and thus suffering the same weakness.

In the original BAN logic, the nonce verification rule is:

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

From this we draw the conclusion that Q believes X, which is a nonce. However, as Mao *et al.* point out, it does not make sense to believe a nonce. One can believe that X is a nonce, or that nonce X is fresh, but one cannot believe a nonce.

Mao and Boyd claim that the method for determining assumptions in a protocol is flawed in the BAN logic. A slight modification to the assumptions could turn a useless protocol into a valuable one or vice versa. Also, there is no way to know if the assumptions are the weakest ones possible. This is obviously desirable.

The last defect of the BAN logic discussed by Mao *et al.* has to do with confidentiality. The authors use the Nessett example [42] to show that a BAN analysis fails to recognize some protocols that give away secrets to attackers. Thus, those pieces of information that must remain secrets must be explicitly designated as such.

The new logic presented by Mao and Boyd requires strict typing of formulas and messages. Thus, a principal may no longer believe a nonce, because a nonce is not of a type that may be believed. In addition, a new idealization process is defined. This process is mostly mechanical, and requires little human intervention. However, as some part of the process requires non-automated judgement, this method still suffers from the very criticism the authors have of the original BAN logic.

In the new idealization, challenges and responses are linked. The human interaction is to determine the types of the various parts of messages, and which responses to associate with the challenges. A new construct, *sup*(Q) is defined to represent the fact that Q is a "super" principal. This means that Q is entirely trusted. An example of this would be an authentication server in the Needham and Schroeder protocol [40]. This is necessary because a principal should not be trusted only on certain beliefs. If a principal is trusted, it must be entirely trusted.

New inference rules are added in the Mao and Boyd logic. They are very similar to rules in the BAN logic, but include the *sup* construct, and are based on the mechanical idealization process. However, the reasoning process is quite different. Mao and Boyd used the *tableau* method [17]. The reasoning starts with the desired conclusion, such as A *believes* A $\stackrel{K}{\leftrightarrow}$ B, and finds rules that lead to that conclusion. Thus, reasoning proceeds backwards until the initial assumptions are found. This method finds the weakest pre-conditions if necessary conditions are always found when searching for rules to apply.

The new logic is applied to the Nessett protocol to uncover the known weakness. Then, a revised version of Otway-Rees is given, and the new logic is used to prove that no conditions are violated when applying rules from the assumptions to the conclusions. Because the logic is not complete, this is the strongest statement that can be made from such a system.

The system described by Mao and Boyd reasons monotonically, as does the BAN logic, so there is no provision for refutation of belief. Human intervention still exists but is much more limited. While this logic is an improvement over previous ones, it does not provide a mechanism for proving that a protocol is correct.

7.3.3 Extending BAN to deal with PKCS

To analyze a public key crypto system (PKCS), the CCITT X.509 Strong Two-Way Authentication Protocol, Gaarder and Snekkenes [18] extend the BAN logic. The following constructs are introduced to represent public key cryptography and time[6]:

$\mathcal{P}K(K,U)$ The entity U has associated the good public key K.

$\Pi(U)$ The entity U has associated some good private key. The key value is only known by U.

$\sigma(X,U)$ The formula X is signed with the private key belonging to U.

$(\Theta(t_1,t_2),X)$ X holds in the interval t_1,t_2. The creator that uttered the time-stamped message X claims that X is good in the time interval between t_1 and t_2.

$\Delta(t_1,t_2)$ The local unique Real Time Clock (RTC) shows a time in the interval between t_1 and t_2.

According to the rules for public key cryptography, we have the rule

$$\frac{U_i \text{ sees } \sigma(X,U_j)}{U_i \text{ sees } X}$$

which states that any principal seeing a formula signed with another principal's private key, can see that formula.

The following rule deals with duration stamps. That is, a formula is good for a certain interval. This is necessary because the CCITT X.509 protocol being analyzed involves the use of time and a real clock.

$$\frac{\text{P } \textit{believes} \text{ Q } \textit{believes} \ \Delta(t_1,t_2), \quad \text{P } \textit{believes} \text{ Q } \textit{said} \ (\Theta(t_1,t_2),X)}{\text{P } \textit{believes} \text{ Q } \textit{believes} \text{ X}}$$

This is similar to one of the rules in the original BAN logic, but it restricts the time during which this rule can be applied to a "good" interval. Calvelli and Varadharajan [9] reason about time in a similar way by introducing a

[6] Recall that the original BAN logic has no provisions for reasoning about time.

rule of the form $(B_i \wedge t)\, says\ r$ that means that B_i *says* r at or before time t.

Gaarder *et al.* also introduce the construct

$$\mathcal{R}(P, X)$$

to say that P is the intended recipient of message X. An example of this would be

$$P \rightarrow Q : \mathcal{R}(Q, X), X$$

where $\mathcal{R}(P, X)$ is a *tag* telling Q that he is the intended recipient of the message.

The authors then proceed to formalize the goals of the protocol, and idealize the protocol. The extended BAN logic is then used to prove that the protocol meets its goals.

7.3.4 Adding probabilistic reasoning to BAN

BAN logic is "...concerned with evaluating the trust that can be put on the goal by the legitimate communicants using beliefs of the principals," according to Campbell *et al.* [10]. It "has no provisions for modeling insecure communication channels or untrustworthy principals and, in fact, fails to model any type of insecurity."

Campbell *et al.* extend the BAN logic using probabilistic reasoning to calculate a *measure* of trust rather than complete trust. That is, given assumptions about the level of trust among principals, and a protocol, we can analyze the level of trust that this protocol achieves.

The authors define the analysis problem in terms of an equivalent linear programming problem as follows:

Let $p_1, ..., p_n$ be an assignment of probabilities to the assumptions $a_1, ..., a_n$ of a proof of the conclusion c. Then, $L \leq P(c) \leq U$, and the lower limit L (respectively upper limit U) can be obtained by solving the linear program:

$$\begin{array}{ll} \text{minimize (resp. maximize)} & z = q \cdot \pi \\ \text{subject to the constraints} & W\pi = p \\ & 1 \cdot \pi = 1, \pi \geq 0 \end{array}$$

The simplex algorithm can then be applied to find a minimal solution. This method is then applied to the Needham and Schroeder protocol. This method easily reveals the known weakness. It also shows that the assumption that K_{ab} is a good key is responsible for this weakness.

The weakness of the Needham and Schroeder protocol was discovered by the extensions of Campbell *et al.*, as the original BAN logic does, without using prior knowledge of it. Although this is a promising result, it does not constitute a proof that the protocol is secure, due to its incompleteness.

One weakness of schemes such as that presented by Campbell *et al.* [10] is difficulty of use. The original BAN logic has been praised for its simplicity, so it seems, and intuition also dictates, that there is a tradeoff between the ease of use of an analysis tool and its utility.

7.4 The CKT5 Logic

Bieber [2] extends the epistemic logic of Hintikka [22]. This new logic of communication in a hostile environment, called CKT5, allows a user to describe the states of knowledge and ignorance associated with the communication via encrypted messages. Bieber also extends the logic of knowledge and time, KT5, of Sato [47] with operators that relate directly to the sending and receiving of messages.[7]

To describe a protocol, P, in CKT5, we define φ to be the way principals behave when participating in P, and ω states who knows what when the protocol terminates. Then, $\varphi \rightarrow \omega$ is a CKT5 formula that describes P. Next, it must be proved that $\varphi \rightarrow \omega$ is a theorem. An example of a member of φ is "if principal A has sent m encrypted under K, then he must have received m at some point in the past." Similarly, ω contains statements such as "At time t, A knows that k is a crypto key."

CKT5 extends the basic epistemic logic by adding modal operators to express the transmission and receipt of messages. The usual connectives are used, $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$, along with the quantifiers, $\forall, \exists$, and equality of terms ($=$). In addition, Bieber defines modal operators $K_{A,t}$, $R_{A,t}$ and $S_{A,t}$. If A and B are principals, t is a number representing time, and φ is a well

[7] The following summary of Bieber's logic borrows examples from Snekkenes [53], who gives an excellent summary of CKT5, and from the original paper by Bieber [2].

formed formula in CKT5, and m, n, m_1, and k are terms, then the following are definitions in CKT5.

$K_{A,t}\varphi$ At time t, A knows that φ holds.

$R_{A,t}\varphi$ At time t, A received some message stating that φ holds.

$S_{A,t}\varphi$ At time t, A sent some message stating that φ holds.

$m.m_1$ The concatenation of m and m_1.

$\mathbf{d}(k, m)$ The decryption of m with key k.

$\mathbf{nonce}(n, t, A)$ n is a nonce generated at time t by the system at the request of A.

$\mathbf{private}(t, \{A, B\}, k)$ k is a symmetric key shared by A and B.

$\mathbf{key}(k)$ k is a symmetric key.

$K_{A,t}\mathbf{msg}(m)$ At time t, A knows that m is a computable term[8].

$\mathbf{belongs}(m_1, m_2)$ m_1 occurs as a subterm of m_2.

$\mathbf{clear}(m)$ m is a clear text message.

$\mathbf{clean}(m)$ m is a clear text message, the concatenation of two clean messages or can be decrypted to yield a clean message.

$\mathbf{n\_s\_clean}(m)$ m contains a subterm that is not necessarily well built[9].

The logic assumes uncertain communication, so that if a message is intercepted by an intruder, it may not arrive at its destination. Messages are not lost, but the intruder can prevent them from reaching a target.

Bieber defines *univoque* messages to be ones that for agent X at time t, are well built Logicwith exclusively clear text messages and keys known only by X at time t. Formally:

$$\begin{aligned}&\text{univoque}(X,t,m) \leftrightarrow K_{X,t}\ \text{clear}(M)\\ &\quad \vee\ (\exists k\ K_{X,t} key(k) \wedge \text{univoque}(X, t, d(k, m)))\\ &\quad \vee\ (\exists m_1\ \exists m_2\ m = m1.m2 \wedge \text{univoque}(X, t, m_1) \wedge\\ &\qquad \text{univoque}(X, t, m_2)\ )\end{aligned}$$

[8] Bieber defines a computable term as a clean message, or a concatenation of two not so clean terms (containing some encrypted part), or the result of performing a cryptographic function on a not so clean term [2].

[9] Bieber defines a well built term as a clear text term, or the concatenation of two clear text terms, or the encryption or decryption of a clear text term [2].

Thus, X knows that a message is usable if it is univoque.

Bieber makes the recommendation that knowledge rather than belief be used to guarantee security because epistemic logics are better at describing the behavior of other agents, as is seen by a strong logic such as CKT5. Syverson makes a similar recommendation [56].

Snekkenes gives an example application of CKT5 [53] with KP, a protocol similar to the Needham and Schroeder protocol. CKT5 is modified so that it can distinguish between the role of a principal and its name. This is done simply by introducing a predicate "role-R" that maps principals to their roles in a protocol.

The proof that KP is secure points out the weakness of CKT5. As it is known that KP has a flaw (discussed in section 3.1), the fact that it can be proved correct in CKT5 demonstrates that strictly epistemic logics, as we know them, are not sufficient for analyzing the security of authentication protocols.

7.5 Analysis of Belief Evolution

An authentication protocol analysis can be viewed as the evolution of the beliefs of the principals involved. Kailar and Gligor [27] present a logic similar to the BAN logic for reasoning about the evolution of belief within a protocol run. The types of protocols that can be analyzed with this logic include interdomain authentication, and protocols where trust in an encryption key must be established despite the lack of jurisdiction.

In this logic, beliefs in a protocol run evolve as in a state machine, where a current belief and an action determine the next state of belief.

$$\text{Belief} + \text{Action} \Rightarrow \text{New Belief}$$

The concept of a *knowledge set* for each message content is introduced. A knowledge set is a set of principals who *know* the contents of a message, a, and a given round in the protocol, M_i. Here, M_i stands for message M at instant i in the protocol run.

A message is represented as:

$$\{Message\ round, Sender, Receiver, Contents\}$$

The sender and receiver field correspond to signing with a given key, or in the case of symmetric keys, to the encryption with a session key. Thus,

$$Y \triangleright \{M_k, Y, X, C\}$$

denotes that Y sends message M with content C in round k of this session to principal X. Similarly,

$$X \triangleleft \{M_k, Y, X, C\}$$

denotes that X sees message M in round k and knows that it is sent by Y. It also reads the message contents C.

Kailar and Gligor's logic represents trust explicitly, thus avoiding some of the problems that arise in the BAN logic based on trust assumptions. The statement $TRUST_R(P,Q)$ means that P trusts principal Q in the context R. Trust means that if Q says X, and P trusts Q, then P believes X. In the analysis, a forwarded message is viewed as being sent from the originator directly to the destination. As the intermediary cannot read the contents, this short circuit makes sense.

One noteworthy assumption made in this logic is that principals can distinguish between messages from a current session, and messages from other sessions. Without this, an inconsistent state of beliefs can be attained due to unrelated message histories.

The logic contains inference rules similar to those of the BAN logic. Most of the rules are concerned with maintaining the knowledge sets, and these sets are what allow principals to reason about the evolution of other principals' beliefs in a protocol. The first inference rule presented is the belief in uniqueness of message receipt. This rule states that if X sends a message, M_i, to Y, and X believes that Z reads the message, then X believes that $Z = Y$.

$$\frac{X \triangleright \{M_i, X, Y, C\}, \quad X \text{ believes } \{Z \triangleleft \{M_i, X, Y, C\}\}}{X \text{ believes } \{Z = Y\}}$$

The following rule defines how knowledge sets are maintained. Here, $KS(a, M_i)$ stands for the knowledge set of contents a of message M_i, and this set contains the principals who know a after having received message M_i. Kailar and Gligor use C to denote the contents of a message, except in the case of knowledge sets where a is used. We follow their conventions.

$$\frac{X \text{ believes } KS(a, M_i) = \{S_1\}; \quad X \text{ believes } KS(a, M_j) = \{S_2\} | j \geq i > 0}{X \text{ believes } \forall Y \in \{S_2\} - \{S_1\}; \ \exists P, k | P \in S; i < k \leq j; Y \triangleleft \{M_k, P, Y, a\}}$$

This states that if the knowledge set for some contents a in a later round number contains principals that are not in the knowledge set of an earlier round number, then those principals must have received a message with that contents during the time interval between the rounds.

Other rules in the logic describe when a formula can be included in a knowledge set, the freshness of nonces, and the freshness of message content. They are similar in style to the ones above and can be found in the original paper [27].

Kailar and Gligor compare the use of their logic to the BAN logic for the analysis of some well known protocols such as an inter-domain authentication protocol, a PROXY ticket forwarding protocol, and a multiparty session protocol. They show that their analysis preserves the accumulation of beliefs of all the principals in the system, whereas the BAN analysis falls a bit short.

Calvelli and Varadharajan [9] use this logic to analyze some delegation protocols for Kerberos, which is evidence of the usefulness of the logic. Others have also used it to analyze new systems. The ease of use of Kailar and Gligor's logic is seen by its applicability to many problems. This advantage is significant as is seen from the complexity and resulting lack of use of methods in the Type IV approach discussed later.

7.6 Semantics of Logics of Authentication

The utility of formal protocol analysis is limited by the quality of the tools we are using. Just as we have formal methods for evaluating protocols, it is useful to be able to reason about the tools themselves.

According to Syverson [57], "One of the main roles of a semantics is to give us a means to evaluate our logics. When evaluating a logic we are primarily interested in two questions: Can we derive everything we want? (Completeness) And, can we avoid deriving things we don't want (Soundness)." In general, we seem to be more concerned with soundness, whereas, for computer security, "completeness is of the utmost importance" [57].

The reason we need to ensure that we can derive anything possible is that many logics rely on the generation of all possible security flaws. If a logic is incomplete, (as is the original BAN logic [7]), then there may be flaws that are overlooked. "A formal semantics provides a precise structure with respect to which soundness and completeness of a logic may be proven" [57]. However, as Syverson explains, the semantics must not be derived directly from the logic. An independently derived semantics for a logic serves as a valuable tool in evaluating the logic.

7.6.1 A semantics for the BAN logic

Abadi and Tuttle [1] define a semantics for the BAN logic. They define belief as a form of resource-bounded defeasible knowledge, using a possible-worlds semantics. First, they remove some unnecessary assumptions in the original assumptions by introducing new constructs. Then, the semantics are formally defined.

The original BAN logic assumes that principals are honest, in that they believe in the truth of the messages they send. To remove this assumption, a new construct is introduced, 'X', which is read "*forwarded* X," that is used for messages that were not constructed by the principal sending them. Another construct introduced before the semantics are defined is "P *says* X" to represent the fact that P has sent X in the present. Using this, a new postulate is introduced that states that if P said X and X is fresh, then P *says* X. This promotes from knowledge to belief.

Another construct deals with shared secrets. If P and Q share a secret, s, then $\langle X^Q \rangle_s$ represents the combination (usually concatenation) of X and s. This is usually used to demonstrate knowledge of a secret.

The BAN logic is reformulated to define the semantics precisely. For the complete description, the reader is referred to the paper [1]. We give a summary of the more important aspects of the semantics. The following actions are defined for a principal P:

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

receive() denotes P's receipt of a message. Some message m is nondeterministically chosen and deleted from P's message buffer.

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

seen-submsgs$_K(M)$ is defined as the union of the set $\{M\}$ and

1. $\textit{seen-submsgs}_K(X_1) \cup \cdots \cup \textit{seen-submsgs}_K(X_k)$if $M = (X_1, \cdots, X_k)$
2. $\textit{seen-submsgs}_K(X)$ if $M = \{X^Q\}_K$ and $K \in \mathcal{K}$[10]
3. $\textit{seen-submsgs}_K(X)$ if $M = \langle X^Q \rangle_s$
4. $\textit{seen-submsgs}_K(X)$ if $M =$ 'X'

said-submsgs$_{K,M}(M)$ This is defined almost the same way as $\textit{seen-submsgs}_K(X)$ except that the fourth condition also stipulates that $X \notin \textit{seen-submsgs}_K(X)$.

Next, Abadi and Tuttle describe the syntactic restrictions on a protocol run, r, a time k, a key set $\mathcal{K}$, a principal P, and M, the set of messages P has received before time k.

1. A principal's key set never decreases: If $\mathcal{K}'$ is P's key set at time $k' \leq k$, then $\mathcal{K}' \subseteq \mathcal{K}$.

2. A message must be sent before it is received: If $\textit{receive}(M)$ appears in p's local history at time k, then $\textit{send}(M, P)$ appears in some principal Q's local history at time k.

3. A principal must possess keys it uses for encryption. Suppose that action $\textit{send}(M, Q)$ appears in P's local history at time k and that $\{X^R\}_K \in \textit{said-submsgs}_{K,M}(M)$. Then, either $\{X^R\}_K \in \textit{seen-submsgs}_K(M)$ or $K \in \mathcal{K}$.

4. A system principal sets "from" fields correctly: if $\textit{send}(M, Q)$ appears in P's local history at time k and $\{X^R\}_K \in \textit{said-submsgs}_{K,M}(M)$, then $P = R$ or

[10] P's key set.

$\{X^R\}_K \in$ *seen-submsgs*$_K(M)$. Similarly, if *send*(M, Q) appears in P's local history at time k and $\langle X^R\rangle_Y \in$ *said-submsgs*$_{K,M}(M)$, then $P = R$ or $\langle X^R\rangle_Y \in$ *seen-submsgs*$_K(M)$.

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

Once the syntax has been defined, the semantics can be described. The definition of $(r, k) \models \varphi$ is inductive on the structure of φ. An interpretation π maps each $p \in \Phi$ to the set of points $\pi(p)$ at which p is true. So,

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

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

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

Next, the semantics are described for the various constructs in the logic. For example P *sees* X at (r, k) is defined as

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

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

- *receive*(M) appears in P's local history
- $X \in$ *seen-submsgs*$_K(M)$, where K is P's key set.

Also, P has jurisdiction over φ at (r, k) is defined as

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

iff $(r, k') \models P$ *says* φ implies $(r, k') \models \varphi$ for all $k' \geq 0$.

The other constructs in the BAN logic are defined similarly in the semantics. The notion of belief is captured using a possible-worlds semantics where a principal believes a fact if that fact is believed in all the possible worlds known to that principal at that time. Abadi and Tuttle [1] prove that this axiomization is sound, but state that they doubt it is complete. They give an example of a formula that is valid, but cannot be generated using the logic:

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

In the semantics described by Abadi and Tuttle, choosing good protocol runs is important. They do not allow an initial assumption with a negative belief, such as P_i does not believe K is a good key. This seems to be a reasonable assumption. As the authors state, "In every application of this logic that we are aware of, the initial assumptions satisfy this restriction."

7.6.2 A semantic model for authentication protocols

Woo and Lam [67] present a semantic model for authentication protocols. They identify *correspondence* and *secrecy* as two correctness properties. Correspondence specifies that different principals in a protocol must execute steps in a locked-step fashion. This represents the idea that a protocol step can be in response to a previous protocol step, and not just an independent event.

The authors define an action schema to specify the steps in a protocol. In protocol specification, each of these actions is preceded by a label. The actions allowed are:

BeginInit (r)	NewNonce (n)
EndInit (r)	NewSecret (S, n)
BeginRespond (i)	Send (p, M)
EndRespond (i)	Receive (p, M)
Accept (N)	GetSecret (n)

The meanings of these actions are for the most part intuitive. A notable exception is the *GetSecret* action. This is used to model the compromise of an old key by the intruder. The action would not be included in a protocol specification, but rather, on the consequence side of a rule, and serves to eliminate timeliness requirements.

A protocol specification begins with a set of initial conditions, followed by the protocol for each participant. For example, the authors specify the Otway-Rees protocol [43] using their model. The specification takes the form.

1. Initial Conditions
2. Initiator Protocol
3. Responder Protocol
4. Server Protocol

Notice that although the Otway-Rees protocol does not differentiate between the communicating principals, Woo and Lam explicitly designate the roles as initiator and responder.

The main difference between this work, and that of Syverson [56] is that Woo and Lam specify protocols as programs and are concerned with a general formalism of correctness, whereas Syverson is more concerned with logic. The approach by Woo *et al.* is revolutionary in that it recognizes and formalizes the notion of correspondence in authentication protocols.

7.7 A Nonmonotonic Logic of Belief

All of the logics we have discussed so far have dealt with monotonic knowledge and belief. However, in a real world model, our beliefs can change. For example, if a session key is compromised, we need to change our belief that this is a good key.

Moser [39] describes a nonmonotonic logic of belief based on a monotonic logic of belief and knowledge. She describes the standard S5 [11] knowledge axioms. Here $K_i(p)$ means that principal i *knows* p.

1. $K_i(p) \Rightarrow p$
2. $K_i(p) \wedge K_i(p \Rightarrow q) \Rightarrow K_i(q)$
3. $\neg K_i(p) \Rightarrow K_i(\neg K_i(p))$ (Negative introspection)
4. $\vdash p$ infer $K_i(p)$

Axiom 4 corresponds to the axiom **T** described by Syverson [61] (see section 7.1). Also, Moser points out that positive introspection is easily derivable from the above axioms. The axioms for belief are the standard KD45 axioms [11], and are the same as those described by Rangan [44] (see section 7.1).

In Moser's logic, a belief is considered true unless it is stated otherwise. She introduces a new predicate, *unless* whose value can be seen from the following truth table (where F is a conjunction of formulas containing the *unless* operator).

The definition of *unless* is given by the truth table in Figure 5. The x in the last row is the most interesting part of the definition and indicates a special case. x is defined as follows:

$B_i(p)$	$B_i(q)$	$B_i(p)$ unless $B_i(q)$
t	t	f
f	t	t
t	f	t
f	f	x

Figure 5: **The Definition of Moser's *unless* operator** The x in the last row indicates a special case. x is true iff $\exists r : B_i(p)$ unless $B_i(r) \in F$, where F is a conjunction of formulas containing the *unless* operator.

$$x = \begin{cases} t & \text{if } \exists r : B_i(p) \text{ unless } B_i(r) \in F \\ & \text{and } B_i(r) \text{ is true} \\ f & \text{otherwise} \end{cases}$$

Thus, the value of the *unless* operator depends on the context in which it appears. This definition allows for any belief to be held unless it is refuted somewhere else in the formula.

Moser proceeds to give an application of this logic for a key distribution protocol. Although her logic provides for a new type of reasoning, there are a few shortcomings. Moser does not discuss the tractability of her logic other than pointing out that if quantification were added to the logic, it would be undecidable. Also, Moser makes no mention of soundness and completeness. Perhaps a formal semantics for this logic would help answer such questions.

Another shortcoming of Moser's logic is that it deals with the nonmonotonicity of belief, and mentions nothing of the nonmonotonicity of knowledge. In fact, there is no way to reason about a principal forgetting some information. An example of such a protocol is the *khat* system [46] of Rubin and Honeyman. The security of *khat* is based on the notion that a vacant workstation erases some information from its memory so that an intruder gains nothing from compromising the machine. To reason about such systems, we need a nonmonotonic logic of knowledge as well as belief. Moser's logic does not provide this.

8 Type IV Approach

The Type IV approach to protocol analysis develops a formal model based on the algebraic term-rewriting properties of cryptographic systems. This approach was introduced by Dolev and Yao [16], and has since been pursued by Syverson [55, 60], Meadows [33, 34, 35, 36], and Woo & Lam [67]. The more recent applications of this approach have provided automated support for the analysis, and have enabled a user to query the system for known attacks.

The Type IV approach generally involves an analysis of the attainability of certain system states. In this regard, it is similar to some of the Type II approaches discussed earlier. However, the Type IV approaches try to show that an insecure state cannot be reached, whereas the Type II approaches began with an insecure state and attempted to show that no path to that state could have originated at an initial state.

8.1 Dolev and Yao

Dolev and Yao [16] proposed the first algebraic model for the security of protocols. Their protocols dealt more with the distribution of secrets than authentication, although the two are closely linked. The main difference is that we generally think of authentication as involving a third party authentication server, whereas the Dolev and Yao protocols dealt with only two parties.

Dolev and Yao define some classes of protocols. They reason about these classes of protocols rather than individual protocols themselves, and prove some interesting properties of these classes. For example, *cascade protocols* and *name-stamp protocols* are examined. A cascade protocol is one in which a user can apply the public key encryption-decryption operations in several layers, to form messages. The authors prove that such protocols are secure if and only if the following conditions hold:

1. The messages transmitted between X and Y always contains some layers of encryption functions E_x or E_y
2. In generating a reply message, each participant A ($A = X, Y$) never applies D_A without also applying E_A.

Similarly, Dolev and Yao provide a polynomial-time algorithm for deciding if a given name-stamp protocol is secure.

Dolev and Yao not only show how to model protocols algebraicly, they consider whole classes of protocols and demonstrate how to reason about any protocol that shares certain properties.

8.2 Using the NARROWER Algorithm for Protocol Analysis

According to Meadows [33], "A cryptographic protocol may be thought of in part as a set of rules for generating words in some formal language." We can define algebraic operations on these words, such as encryption and decryption. The security of a protocol can then be based on the ability of an intruder to generate certain words in the language.

The operations in a term-rewriting system are the reductions of terms using the cancellation properties of the words in the system. Examples of such rules are:

1. $d(X, e(X, Y)) \rightarrow Y$
2. $e(X, d(X, Y)) \rightarrow Y$

which define the symmetric properties of encryption and decryption. An intruder can attempt to see if any words available to him can reduce to some secret word, say a session key.

Meadows [33] uses the NARROWER [45] algorithm, which she has implemented in Prolog [13] to analyze the IBM key management scheme [14] mentioned in Section 6.2. The algorithm begins with a trivial set (possibly the empty set) of words available to the intruder, and an initial state. A set of secrets, which the intruder should not learn, is also defined. The algorithm attempts to show that there is no path through the protocol, beginning at the initial state, that leads to a state where the intruder can learn words in the secret set.

The algorithm works by induction on the length of the path, beginning with the trivial set, and continues until no more paths can be generated. For any m, we can state that no "dangerous" path of length less than or equal to m exists. The user can interact with the program to improve on the tractability of the problem by modifying the initial set, and the rules available.

In a later paper [35], Meadows analyzes the Burns-Mitchell resource sharing protocol [6] using an analysis tool based on the same term-rewriting properties utilized by the NARROWER. This system models the knowledge and belief of the intruder, and defines a set of rules whereby an intruder can learn new information based on protocol steps. Using this technique, Meadows demonstrates the existence of a flaw in the Burns-Mitchell protocol. Meadows suggests a fix to this protocol, and then uses the analysis technique to show that the attack no longer succeeds.

In the analysis of the IBM key management scheme, it is shown that certain secrets are unobtainable by a penetrator unless a session key has been compromised. However, such a proof is not a proof that the protocol is *secure*; this would require proving that the term-rewriting system is complete, *i.e.*, that every valid word can be generated. Unfortunately, the system is not complete. Another requirement for proving that a protocol is secure is that the method for formalizing the protocol, must itself be formal, and it is not.

Thus, methods such as using the NARROWER can be used to find insecurities in a protocol, but do not constitute a proof that a protocol is secure.

8.3 The KPL Logic

In the logics presented so far, we have seen ways of representing the fact that P knows that K_{PQ} is the secret key between P and Q. However, there has been no way to represent simply the fact that intruder Z knows P's key. Syverson calls this a key *simpliciter*, and his KPL logic [55, 56] can represent such a fact.

KPL is a quantified modal logic with a corresponding possible-worlds interpretation. In KPL, Z knows P's key if P's key is present in all of the worlds accessible to Z from his current set of possible worlds via some transition. Syverson defines a semantics for the logic that he uses to prove the soundness and completeness of KPL.

As Syverson states, the soundness and completeness of KPL do not guarantee that there can be no error in the reasoning about a secure protocol, but they do prove that there can be no formal error: "Once we have formally specified a protocol, a logical derivation of any result concerning the specification will be correct — *i.e.* true of that specification — and anything that can be formally shown to be a semantic consequence of that specification will be provable in the logic."

Syverson does not provide examples of how to specify an authentication protocol in KPL; such a specification would be complicated. In general, the Type IV approaches suffer from a great deal of complexity, and thus their value as an analysis tool is diminished.

8.4 The NRL Protocol Analyzer

Syverson and Meadows [60] use the techniques described above, namely, using the term-rewriting properties of protocol specifications, to develop the NRL protocol analyzer. This system is used to analyze classes of protocols, and is not tied to any assumptions about the protocols.

The NRL protocol analyzer allows a single set of requirements to specify a class of protocols. The following symbols are used:

$\rightarrow$ represents the standard conditional

$\wedge$ represents conjunction

$\diamondminus$ represents a temporal operator meaning *at some point in the past.*

Each principal keeps track of a local round number for a protocol, and the following actions are defined:

accept(B, A, Mes, N) means that B accepts the message Mes as from A during B's local round N. N is an optional parameter.

learn(Z, Mes) means that the intruder, Z, learns the word Mes.

send$(A, B, (Query, Mes))$ means that A sends Mes to B in response to $Query$. The use of a $Query$ is optional.

request$(B, A, Query, N)$ means that B sends $Query$ to A during B's local round N. $Query$ is optional.

From these constructs and actions, requirements can be specified. The requirements are represented by a conjunction of statements. For example:

Requirement #1

- $\neg(\diamond\, \text{accept}(B,A,Mes,N) \wedge \diamond\, \text{learn}(Z,Mes))$
- $\text{accept}\ (B,A,Mes,N) \rightarrow$
  $\diamond\, (\text{send}(A,B,(Query,Mes)) \wedge$
  $\diamond\, \text{request}(B,A,Query,N))$

Requirement #1 contains two conditions, both of which must hold. The first condition is that if B accepted message *Mess* from A at some point in the past, then the intruder did not learn *Mess* at some point in the past. The second condition is that if B accepted message *Mess* from A in B's local round, N, then A sent $Mess$ to B as a response to a query at B's local round N.

It is clear from the last sentence why formal methods are needed to represent such statements. There is a need for a precise description. If it is not necessary for A to send the message in response to B's query *only* after the query, then we can have the relaxed requirement:

Requirement #2

- $\neg(\diamond\, \text{accept}(B,A,Mes) \wedge \diamond\, \text{learn}(Z,Mes))$
- $\text{accept}\ (B,A,Mes,N) \rightarrow$
  $\diamond\, (\text{send}(A,B,Mes)) \wedge \diamond\, \text{request}(B,A,N))$

The omission of the *Query* from the *send* and *request* actions are due to the relaxation of the requirement. Also, it may be required that the messages from A and B be recent. These can be specified with the following requirement:

Requirement #3

- $\neg(\diamond\, \text{accept}(B,A,Mes) \wedge \diamond\, \text{learn}(Z,Mes))$
- $\text{accept}\ (B,A,Mes,N) \rightarrow$
  $\diamond\, (\text{send}(A,B,Mes)) \wedge \diamond\, \text{request}(B,A,N))$
- $\text{accept}\ (B,A,Mes,N) \rightarrow$
  $\diamond\, (\text{send}(A,B,Mes)) \wedge$
  $\neg(\diamond\, \text{time\_out}(B,N)) \wedge$
  $\neg(\diamond\, \text{time\_out}(A,N))$

The new action, *time_out*, is used to control currency of messages.

To avoid assumptions such as those made in the BAN logic [7] that all participants in a protocol are honest, an honest user A is designated as $user(A, honest)$, and a dishonest user as $user(A, dishonest)$, and in the case where it is not known, a variable Y can be included, $user(A,Y)$. To specify the requirement that an honest B accept a message as coming from an honest A only if it was never previously accepted by an honest user, Syverson and Meadows [60] provide the following complicated requirement:

Requirement #4

- $\neg\diamond\, \text{accept}(user(B,honest), user(A,honest),$
  $Mes) \vee \neg\diamond\, \text{learn}(Z,Mes)$
- $\text{accept}(user(B,honest), user(A,honest), Mes, N)$
  $\rightarrow \diamond\, (\text{send}\ (user(A,honest), user(B,honest), Mes)$
  $\wedge\diamond\, \text{request}\ (user(B,honest), user(A,honest), N))$
- $\text{accept}(user(B,honest), user(A,honest), Mes) \rightarrow$
  $\neg\diamond\, \text{accept}(user(C,honest), user(D,Y), Mes)$

The variable Y states that the message could not have been accepted by anyone, regardless of honesty.

The NRL protocol consists of a state space, where a protocol trace is an infinite sequence of states. A *model* is an ordered 4-tuple, $\langle S, I, \sigma, t\rangle$ such that S is a state space, I is an interpretation, σ is a trace, and t is time. Satisfaction is defined as $\langle S, I, \sigma, t\rangle \models \alpha$ means that α is true at $\langle S, I, \sigma, t\rangle$. Syverson and Meadows [60] give a detailed definition of the $\models$ relation in their paper.

A specification in the NRL protocol analyzer is modeled on the communication of honest participants. Dishonest participants are assumed to be modeled by the intruder, and so are not modeled separately. Each honest participant possesses a set of learned facts, called *lfacts*. Also, a function called *intruderknows()* represents the lfacts known by the intruder.

To use the protocol analyzer, a protocol is specified using the above constructs. Then, a set, K, of lfacts is defined. K may contain a possible attack by an intruder. The analyzer then can determine if the set K can meet the requirements of the protocol. If so, a successful attack by the intruder can be discovered.

The NRL protocol analyzer consists of four phases. In the first phase, transition rules are defined for the actions of honest principals. In phase 2, the operations available to all principals, such as encryption and decryption, are described. Phase 3 consists of describing the atoms used as the basic building blocks of the words in the protocol, and the final phase describes the rewrite (reduction) rules obeyed by the operators. An example of such a rule is:

IF:
  count(user(B,honest)) = [M],
  lfact(user(B,honest),N,recwho,[11] M) =
    [user(A, Y)],
  not(user(A, Y) = user(B,honest)),
THEN
  count(user(B,honest)) = [$s(M)$],
  intruderlearns([user(B,honest),
    rand(user(B,honest),M)]),
  lfact(user(B,honest),N,recsendsnonce,
    $s(M)$) = [rand(user(B,honest),M)],
EVENT
  event(user(B,honest),N,requestedmessage,
    $s(M)$) = [user(A,Y),rand(user(B,honest),M)].

This rule describes the sending of a nonce from an honest principal, B, to some principal A, whose honesty is unknown.

Operations are described by listing the restrictions on them (for example, key length), and then defining their properties. Similarly, rewrite rules describe the cancellation of operations. For public key encryption, for example, a rewrite rule would be:

$$\text{pke}(\text{privkey}(X),\text{pke}(\text{pubkey}(X), Y)) \Rightarrow Y$$

Syverson *et al.* [60] then give a full specification and show that the protocol meets the specification.

The purpose of the NRL protocol analyzer is to show that a given protocol specification meets its requirements. However, this does not constitute a proof that the protocol is secure. The NRL protocol analyzer can be viewed as a tool that, combined with other tools, could eventually lead to a protocol that can be proven secure.

9 Conclusions

This paper surveys the current state of research into the formal analysis of authentication protocols. The field has made substantial progress in the detection of flaws in published protocols as well as in the development of formal specification techniques. We have seen how various techniques from other fields can be used to reason about security in key management schemes. We have also seen the weaknesses of such methods in that they fail to capture the subtle properties of these protocols, such as their susceptibility to replay attacks.

Some authors have developed expert systems to experiment with various scenarios in an authentication protocol. Such systems are useful as tools in the development of protocols, but have not been able to offer much in the way of formal analysis of existing protocols. In particular, such systems are good at recognizing known attacks on protocols when they are specified, but have not been able to produce new, previously unknown attacks.

The predominant technique for analyzing cryptographic protocols is to use logical reasoning about belief and knowledge in the system. These schemes have been successful in proving that a protocol meets its formal requirements. However, a criticism of these systems is that the process of formalizing the requirements is itself not formal. To cope with this, semantics have been presented for reasoning about the logics themselves. There is a debate as to whether epistemic logics (knowledge) are preferable to doxastic logics (belief), but either one can be used to reason about the other.

Another criticism of logics based on belief and knowledge is that they are used to model trust and not security. Although the two are related, it is clear that they are not interchangeable.

In addition to the above methods, some have used the algebraic term-rewriting properties of protocols to reason about security. This technique has been successful in uncovering flaws in known protocols. Unlike the modeling with belief and knowledge, the term-rewriting algebras are highly complex. It it doubtful that many protocol developers will be able to use these systems. In contrast, it is common to find analyses of protocols using BAN. However, the ease of use of techniques such as BAN creates a danger of misuse by people who do not fully understand their purpose and limitations, as has been frequently demonstrated.

Although we have presented numerous ways to reason about the security of protocols, and in some cases, to prove that they meet their

[11] The word *recwho* is used by the authors to mean "the principal who receives this."

requirements, there is no technique known for proving that a protocol is secure. The reason for this may be that security itself is not sufficiently well defined. We can prove that a protocol is correct, or that it meets its specification. We can even prove that under various assumptions, certain attacks against a protocol will not work. However, we have no general-purpose method of proving that an arbitrary authentication protocol is secure.

Future research is likely to focus on formal methods for formalizing authentication protocols. The weakest link in current proofs of security is the formalization process. We believe that once all of the aspects of a protocol can be converted to a formal specification using a sound and complete formal method, that we will then be able to assure a proven level of security.

Acknowledgements

The authors thank Atul Prakash for his excellent advice to pursue this topic; Paul Syverson for providing valuable reference materials; and Mary Jane Northrop for helpful comments and editing. This work was partially funded by IBM.

References

[1] Martin Abadi and Mark R. Tuttle. A semantics for a logic of authentication. *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, August 1991.

[2] P. Bieber. A logic of communication in a hostile environment. *Proceedings of the Computer Security Foundation Workshop III*, pages 14–22, June 1990.

[3] Thomas Blumer and Deepinder P. Sidhu. Mechanical verification and automatic implementation of communication protocols. *IEEE Transactions on Software Engineering*, SE-12(8):827–843, August 1986.

[4] D.E. Britton. Formal verification of a secure network with end-to-end encryption. *Proceedings of the 1984 Symposium on Security and Privacy*, pages 154–166, May 1984.

[5] L. Brownston and E. Kant. *Programming Expert Systems in OPS5*. Addison Wesley, 1985.

[6] J. Burns and C. J. Mitchell. A security scheme for resource sharing over a network. *Computers and Security*, 9:67–76, February 1990.

[7] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8, February 1990.

[8] M. Burrows, M. Abadi, and R. Needham. Rejoinder to Nessett. *Operating System Review*, 24(2):39–40, April 1990.

[9] Claudio Calvelli and Vijay Varadharajan. An analysis of some delegation protocols for distributed systems. *Proceedings of the Computer Security Foundationn Workshop V*, pages 92–110, 1992.

[10] E. A. Campbell, R. Safavi-Naini, and P. A. Pleasants. Partial belief and probabilistic reasoning in the analysis of secure protocols. In *Proceedings of the Computer Security Foundationn Workshop V*, pages 84–91, Washington, 1992.

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

[12] P. C. Cheng and V.D. Gligor. On the formal specification and verification of a multiparty session protocol. *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 216–233, May 1990.

[13] W. F. Clocksin and C. S. Mellish. *Programming in Prolog*. Springer-Verlag, 1984.

[14] D. W. Davies and W.L. Price. *Security for Computer Networks*. Wiley, 1984.

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

[16] D. Dolev and A. Yao. On the security of public-key protocols. *Communications of the ACM*, 29:198–208, August 1983.

[17] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. D. Reidel Publishing Company, 1983.

[18] Klaus Gaarder and Einar Snekkenes. Applying a formal analysis technique to the CCITT X.509 strong two-way authentication protocol. *Journal of Cryptology*, 3:81–98, 1991.

[19] Li Gong. Handling infeasible specifications of cryptographic protocols. *Proceedings of the Computer Security Foundationn Workshop IV*, pages 99–102, 1991.

[20] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. *Proceedings of the IEEE Computer Society Symposium on Security and Privacy*, pages 234–248, May 1990.

[21] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. *Proceedings of the Eighteenth ACM Symposium on the Theory of Computing*, pages 304–415, 1986.

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

[23] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[24] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.

[25] J.H. Howard, M.L. Kazar, S.G. Menees, D.A. Nichols, M. Satyanarayanan, R.N. Sidebotham, and M.J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6(1):51–81, February 1988.

[26] James W. Gray III and Paul F. Syverson. A logical approach to multilelvel security of probabilistic systems. *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 164–176, May 1992.

[27] R. Kailar and V. D. Gilgor. On belief evolution in authentication protocols. *Proceedings of the Computer Security Foundation Workshop IV*, pages 103–116, June 1991.

[28] T. Kasami, S. Yamamura, and K. Mori. A key management scheme for end-to-end encryption and a formal verification of its security. *Systems, Computers, Control*, 13:59–69, May-June 1982.

[29] Richard A. Kemmerer. Analyzing encryption protocols using formal verification techniques. *IEEE Journal on Selected areas in Communications*, 7(4):448–457, May 1989.

[30] D. Longley and S. Rigby. Use of expert systems in the analysis of key management systems. *Security and Protection in Information Systems*, pages 213–224, 1989.

[31] W. P. Lu and M. K. Sundareshan. Secure communication in Internet environments: A hierarchical key management scheme for end-to-end encryption. *IEEE Transactions on Communications*, 37(10):1014–1023, October 1989.

[32] Wenbo Mao and Colin Boyd. Towards formal analysis of security protocols. *Proceedings of the Computer Security Foundationn Workshop VI*, pages 147–158, June 1993.

[33] Catherine Meadows. Using narrowing in the analysis of key management protocols. *Proceedings of the 1989 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 138–147, May 1989.

[34] Catherine Meadows. Representing partial knowledge in an algebraic security model. *Proceedings of the Computer Security Foundation Workshop III*, pages 23–31, June 1990.

[35] Catherine Meadows. A system for the specification and analysis of key management protocols. *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 182–195, May 1991.

[36] Catherine Meadows. Applying formal methods to the analysis of a key management protocol. *Journal of Computer Security*, 1(1):5–35, 1992.

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

[38] Judy H. Moore. Protocol failures in cryptosystems. *Proceedings of the IEEE*, 76(5):594–602, May 1988.

[39] Louise E. Moser. A logic of knowledge and belief for reasoning about computer security. *Proceedings of the Computer Security Foundation Workshop II*, pages 57–63, 1989.

[40] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[41] R.M. Needham and M.D. Schroeder. Authentication revisited. *Operating Systems Review*, 21:7, January 1987.

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

[43] D. Otway and O. Rees. Efficient and timely mutual authentication. *ACM Operating System Review*, 21(1):8–10, January 1987.

[44] P. V. Rangan. An axiomatic basis of turst in distrbiuted systems. *Proceedings of the 1988 Symposium on Security and Privacy*, pages 204–211, May 1988.

[45] P. Rety, C. Kirchner, H. Kirchner, and P. Lescanne. Narrower: a new algorithm for unification and its application to logic programming. *Rewriting Techniques and Applications, Lecture Notes in COmputer Science*, 202, 1985.

[46] A. D. Rubin and P. Honeyman. Long running jobs in an authenticated environment. *USENIX Security Conference IV*, pages 19–28, October 1993.

[47] M. Sato. Study of kripke-style models of some modal logics by gentzen's sequential method. *Publications of the Research Institute for Mathematical Sciences*, 13(2), 1977.

[48] J. Scheid and S. Holtsberg. *Ina Jo Specification Language Reference Manual.* Systems Development Group, Unisys Corporation, September 1988.

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

[50] Deepinder P. Sidhu. Authentication protocols for computer networks: I. *Computer Networks and ISDN Systems*, 11:297–310, 1986.

[51] Einar Snekkenes. Authentication in open systems. *Proceedings of the IFIP WG 6.1 Tenth International Symposium on Protocol Specification, Testing, and Verification*, pages 311–324, June 1990.

[52] Einar Snekkenes. Exploring the ban approach to protocol analysis. *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 171–181, May 1991.

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

[54] J.G. Steiner, B.C. Neuman, and J.I. Schiller. Kerberos: An authentication service for open network systems. In *Usenix Conference Proceedings*, pages 191–202, Dallas, Texas, February 1988.

[55] Paul Syverson. A logic for the analysis of cryptographic protocols. Technical Report 9305, Naval Research Laboratory, December 19.

[56] Paul Syverson. Formal semantics for logics of cryptographic protocols. *Proceedings of the Computer Security Foundation Workshop III*, pages 32–41, June 1990.

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

[58] Paul Syverson. Adding time to a logic of authentication, 1993. to appear in ACM Computer Conference, Wahsington.

[59] Paul Syverson. On a key distribution protocol of Newman and Stubblebine. *Submitted to Operating System Review*, 1993.

[60] Paul Syverson and Catherine Meadows. A logical language for specifying cryptographic protocol requirements. *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 165–177, May 1993.

[61] Paul F. Syverson. Knowledge, belief, and semantics in the analysis of cryptographic protocols. *Journal of Computer Security*, 1:317–334, 1992.

[62] V. Varadharajan and S. Black. Formal specification of a secure distributed system. *Proceedings of the 12th National Computer Security Conference*, pages 146–171, October 1989.

[63] Vijay Varadharajan. Verification of network security protocols. *Computers and Security*, 8(8):693–708, 1989.

[64] Vijay Varadharajan. Use of a formal description technique in the specification of authentication protocols. *Computer Standards and Interfaces*, 9:203–215, 1990.

[65] V. L. Voydock and S. T. Kent. Security mechanisms in high–level network protocols. *Computing Surveys*, 15(2):135–171, June 1983.

[66] C. H. West. General technique for communications protocol validation. *IBM Journal of Research and Development*, 22:393–404, 1978.

[67] Thomas Y.C. Woo and Siman S. Lam. A semantic model for authentication protocols. *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 178–194, May 1993.