Technical Report

UCAM-CL-TR-138
ISSN 1476-2986

Number 138

Authentication: a practical study in belief and action

Michael Burrows, Martín Abadi,
Roger Needham

June 1988

ISSN 1476-2986

Authentication: A Practical Study in Belief and Action

Michael Burrows, Martín Abadi, and Roger Needham

Digital Equipment Corporation
Systems Research Center

Questions of belief and action are essential in the analysis of protocols for the authentication of principals in distributed computing systems. In this paper we motivate, set out, and exemplify a logic specifically designed for this analysis; we show how protocols differ subtly with respect to the required initial assumptions of the participants and their final beliefs. Our formalism has enabled us to isolate and express these differences in a way that was not previously possible, and it has drawn attention to features of the protocols of which we were previously unaware. The reasoning about particular protocols has been mechanically verified.

This paper starts with an informal account of the problem, goes on to explain the formalism to be used, and gives examples of its application to real protocols from the literature. The final sections deal with a formal semantics of the logic and conclusions.

The permanent affiliation of M. Burrows and R. Needham is with the Computer Laboratory, Cambridge University. This is a slight revision of a paper published in the Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge, held in Asilomar in February of 1988.

The Problem

In distributed computing systems and similar networks of computers it is necessary to have procedures by which various pairs of *principals* (people, computers, services) satisfy themselves mutually about each other's identity. A common way to approach this is by means of shared secrets, usually encryption keys. In barest outline an *authentication protocol* guarantees that if the principals really are who they say they are then they will end up in possession of one or more shared secrets (e.g., [NS1]). The protocols we are considering here use an *authentication server* that shares a secret with each principal (rather like knowing a password). An authentication server is trusted to make proper use of the data it contains when executing authentication protocols; authentication servers are often also trusted to generate new shared secrets in a proper manner, for example not to issue the same one every time.

Authentication would be straightforward in a sufficiently benign environment. Such cannot usually be assumed, and it is particularly necessary to take precautions against confusion being caused by the reissue of old messages, specifically avoiding the possibility of replays forcing the use of an old and possibly compromised shared secret. A good part of the work in the literature is devoted to ensuring that the information upon which the protocols act is timely. The style of the precautions taken has caused it to be recognized for a long time that we are dealing with questions of belief, trust, and delegation; however this has been recognized imprecisely rather than in any formalism.

As a result of varied design decisions appropriate to different circumstances, a variety of protocols exist and there is a perceived need to be able to explicate them formally so as to be able to understand to what extent they really achieve the same results. There are in fact subtle differences in their final states, and sometimes a protocol may be shown to depend on assumptions which one might not care to make. The purpose of our logic of belief and action is to make us able to explain the protocols step by step, with all assumptions made explicit and with the final states clearly set out.

It is important to note that certain aspects of authentication protocols have been deliberately ignored in our treatment. In particular, there is no attempt to deal with the authentication of an untrustworthy principal, nor to detect unauthorized release of shared secrets via covert channels or weaknesses of encryption schemes (as in [MCF]). Rather, our study concentrates on the beliefs of the parties involved in the protocols and on the evolution of these beliefs as a consequence of communication. Our previous experience with these protocols has indicated that this kind of study is one of the most needed by current protocol designers.

Before introducing the notation and technical terminology, it may be worth saying the main principles in the vernacular. They cannot be regarded as precise, of course:

> If you believe that only you and Joe know K, then you ought to believe that anything you receive encrypted with K as key comes originally from Joe.
>
> If you've sent Joe a number which you have never used for this purpose before and subsequently receive from Joe (that is, encrypted with K) something that depends on knowing that number then you ought to believe that Joe's message originated

recently—in fact after yours.

If both you and Joe believe that the same K is a suitable and timely shared secret, then you've completed authentication.

The Formalism

Authentication protocols are typically described by listing the messages sent between the principals, symbolically showing the contents of each message, the source, and the destination. This conventional notation is not convenient for manipulation in a logic, since we wish to attach exact meanings to each part of each message and these meanings are not always obvious from the data contained in the messages. In order to introduce a more useful notation whilst preserving correspondence with the original description of the protocols, we transform each message into a logical formula. This formula is an idealized version of the original message. Then we annotate each idealized protocol with assertions, much as in a proof in Hoare logic ([H]). These assertions are expressed in the same notation used to write messages. An assertion usually describes beliefs held by the principals at the point in the protocol where the assertion is inserted.

In this Section, we describe the informal syntax and semantics of our logic, its rules of inference, the transformations that we apply to protocols before their formal analysis, and the rules to annotate protocols.

Basic notation

Our formalism is built on a many-sorted modal logic. In the logic, we distinguish several types of objects: principals, encryption keys, and messages. We identify messages with statements in the logic. Typically, the symbols A, B, and S denote specific principals; K_{ab}, K_{as}, and K_{bs} denote specific keys; N_a, N_b, and N_c denote specific statements. The symbols P and Q range over principals; X and Y range over statements; K ranges over encryption keys. All these may be used as either metasymbols (to write schemata) or as free variables (with an implicit universal quantification); this minor confusion is essentially harmless.

The only propositional connective is conjunction, denoted by a comma. Throughout, we treat conjunctions as sets and take for granted their properties. In addition to conjunction, we use the following constructs:

$P \mid\equiv X$: *P believes X*, or P would be entitled to believe X. In particular, the principal P may act as though X is true.

$P \mid\sim X$: *P once said X*. The principal P at some time sent a message including the statement X. It is not known whether the message was sent long ago or during the current run, but it is known that P believed X when he sent the message.

$P \Rightarrow X$: P has *jurisdiction* over X. The principal P is an authority on X and should be trusted on this matter, should he make his opinion known. This construct is used when a principal has delegated authority over some statement or set of statements. For example, encryption keys need to be generated with some care, and in some protocols servers are trusted to do this properly. This may be expressed by the assumption that the principals believe that the server has jurisdiction over statements about the quality of keys.

$P \overset{K}{\leftrightarrow} Q$: P and Q may properly use the *good key* K to communicate. The key K will never be discovered by any principal except P or Q, or a principal trusted by both P and Q.

$\{X\}_K$: This represents the formula X encrypted under the key K. In actuality, $\{X\}_K$ is an abbreviation for an expression of the form $\{X\}_K$ *signed* P. The signature of a particular principal P appears when encryption is used, but this signature is unreadable by everyone but P, hence it is of little importance in authentication and we typically omit it. (Signatures are justified by the realistic assumption that a host can always identify his own messages.)

$P \triangleleft X$: *P sees X*. The principal P has received a message containing X and P can read X (possibly after doing some decryption). This implies that P is able to repeat X in other messages.

$\sharp(X)$: 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 usually true for *nonces*, that is, expressions invented for the purpose of being fresh. Nonces commonly include a number that is used only once.

A formal treatment of the semantics of these constructs can be found below. Here, we simply give and motivate the rules of inference that characterize them.

Logical postulates

Some informal preliminaries are useful to understand the rules of inference of our logic.

In the study of authentication, we are concerned with the distinction between two epochs: the *past* and the *present*. The present epoch begins at the start of the particular run of the protocol under consideration. All messages sent before this time are considered to be in the past, and the authentication protocol should be careful to prevent any such messages from being accepted as recent. All beliefs held in the present are stable for the entirety of the protocol run; furthermore, we assume that when principal P says X then he actually believes X. However, beliefs held in the past are not necessarily carried forward into the present. The simple division of time into past and present suffices for our purposes.

An encrypted message is represented as a logical statement bound together and encrypted with an encryption key. It is assumed that the encryption is done in such a way that we know the whole message was sent at once. A message cannot be understood by a principal that does not know the key; the key cannot be deduced from the encrypted message. Each encrypted message contains sufficient redundancy to be recognized as such and to be decrypted unambiguously. In addition, messages contain sufficient information for a principal to detect (and ignore) his own messages.

Now we are ready to discuss the logical postulates. We do not present the postulates in their most general form; our main concern is to have enough machinery to carry out some realistic examples and to explain the essence of our method.

- The *message meaning* rule concerns the interpretation of encrypted messages:

$$\frac{P \models Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \models Q \mid\!\sim X}$$

Recall that $\{X\}_K$ stands for a formula of the form $\{X\}_K$ *signed* R. As a side condition, it is required that $R \neq P$, that is, $\{X\}_K$ is not "signed" by P himself.
In real life the decryption of a message to yield a content says, in and of itself, only that the content was produced at some time in the past; we have no idea whether it is new or the result of a replay. Thus, the rule states that if P believes that the key K is shared with Q and sees a message X encrypted under K, then P believes that Q once said X.

- The *nonce verification* rule expresses the check that a message is recent, and hence that the sender still believes in it:

$$\frac{P \models \sharp(X), \quad P \models Q \mid\!\sim X}{P \models Q \models X}$$

That is, if P believes that X could only have been uttered recently and that Q once said X, then P believes that Q has said X recently, and hence that Q believes X. For the sake of simplicity, X must be "cleartext," that is, it should not include any subformula of the form $\{Y\}_K$. (This restriction may motivate the introduction of a "has recently said" operator, that we do not currently need.)
This is the only postulate that promotes from $\mid\!\sim$ to $\models$. It reflects in an abstract and timeless way the practice of protocol designers of using challenges and responses. One participant issues a fresh statement as a challenge. Since the challenge has been generated recently, any message containing it is accepted as timely and taken seriously. In general, challenges need not be encrypted but responses must be.

- The *jurisdiction* rule states that if P believes that Q has jurisdiction over X then P trusts Q on the truth of X:

$$\frac{P \models Q \Rightarrow X, \quad P \models Q \models X}{P \models X}$$

- The only remaining necessary property of the belief operator is that P believes a set of statements if and only if P believes each individual statement separately. This justifies the following rules:

$$\frac{P \models X, \quad P \models Y}{P \models (X,Y)} \qquad \frac{P \models (X,Y)}{P \models X} \qquad \frac{P \models Q \models (X,Y)}{P \models Q \models X}$$

- A similar rule applies to the operator $\mid\!\sim$:

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

- The same key is used between a pair of principals in either direction. We use the following two rules to reflect this property:

$$\frac{P \models R \overset{K}{\leftrightarrow} R'}{P \models R' \overset{K}{\leftrightarrow} R} \qquad \frac{P \models Q \models R \overset{K}{\leftrightarrow} R'}{P \models Q \models R' \overset{K}{\leftrightarrow} R}$$

- If a principal sees a formula then he also sees its components, provided he knows the necessary keys:

$$\frac{P \triangleleft (X, Y)}{P \triangleleft X} \qquad \frac{P \models Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \triangleleft X}$$

- If one part of a message is known to be fresh, then the entire message must also be fresh:

$$\frac{P \models \sharp(X)}{P \models \sharp(X, Y)}$$

  Other similar rules can be written, for instance to show that if X is fresh then $P \mid\!\sim X$ is fresh; we do not need these rules in our examples.

Given the postulates, we can construct proofs in the logic. A formula X is provable in the logic from a formula Y if there is a sequence of formulas $Z_0, \ldots, Z_n$ where $Z_0 = Y$, $Z_n = X$, and each Z_{i+1} can be obtained from previous ones by the application of a rule. As usual, this can be generalized to prove schemata.

Idealized protocols

In the literature, each protocol step is typically written in the form

$$P \rightarrow Q : \mathit{message}.$$

This denotes that the principal P sends the *message* and that the principal Q receives it. We transform each protocol step into an idealized form, which expresses logically the semantics of the message. A message in the idealized protocol is a formula. For instance, the protocol step

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

may tell B, who knows the key K_{bs}, that K_{ab} is a key to communicate with A. This step might be idealized as

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

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

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

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

We do not include some cleartext message parts in idealized protocols; most idealized messages are of the form $\{X_1\}_{K_1}, \ldots, \{X_n\}_{K_n}$. We have omitted cleartext communication simply because it can be forged, and so its contribution to an authentication protocol is mostly one of providing hints as to what might be placed in encrypted messages. We cannot reason about the origin or veracity of cleartext.

In general, it is easy to derive a useful logical form for a protocol once it is intuitively understood. However, the idealized form of each message cannot be determined by looking merely at a single protocol step by itself. Only knowledge of the entire protocol can determine the essential logical contents of the message. There are guidelines to control what transformations are possible, and these help in determining the idealized form for a particular protocol step. Roughly, a real message m can be interpreted as a formula X if whenever the recipient gets m he may deduce that the sender must have believed X when he sent m. Real nonces are transformed into arbitrary new formulas; throughout, we assume that the sender believes these formulas. Most important, for the sake of soundness, we always want to guarantee that each principal believes the formulas that he generates as messages.

Protocol analysis

From a practical viewpoint, the analysis of a protocol is performed as follows:

- The idealized protocol is derived from the original one.
- Assumptions about the initial state are written.
- Logical formulas are attached to the statements of the protocol, as assertions about the state of the system after each statement.
- The logical postulates are repeatedly applied to the assumptions and the assertions, in order to discover the beliefs held by the parties in the protocol.

This procedure may be repeated as new assumptions are found to be necessary and as the idealized protocol is refined.

More precisely, we annotate idealized protocols with formulas and manipulate these formulas with the postulates. A protocol is a sequence of "send" statements $S_1, \ldots, S_n$. An *annotation* for a protocol consists of a sequence of assertions inserted before the first statement and after each statement; the assertions we use are conjunctions of formulas of the forms $P \models X$ and $P \triangleleft X$. The first assertion contains the assumptions, while the last assertion contains the conclusions. Roughly, annotations can be understood as simple formulas in Hoare logic. We write them in the form

$$[\mathit{assumptions}]S_1[\mathit{assertion}\ 1]\ldots[\mathit{assertion}\ n-1]S_n[\mathit{conclusions}].$$

(In the examples below, however, we do not demonstrate the use of this notation, to concentrate on the assumptions and conclusions.)

We want annotations to be valid in the following sense: if the assumptions hold initially then each assertion holds after the execution of the protocol prefix that it follows. Clearly, validity is a semantic concept. Its syntactic counterpart is derivability; we give rules to derive legal annotations to protocols:

- For single protocol steps: the annotation $[Y]\,(P \rightarrow Q : X)\,[Y,\, Q \triangleleft X]$ is legal. All formulas that hold before a message is sent still hold afterwards; the only new development is that the recipient sees the message.
- For sequences of protocol steps: if $[X]S_1 \ldots [Y]$ and $[Y]S'_1 \ldots [Z]$ are legal then so is $[X]S_1 \ldots [Y]S'_1 \ldots [Z]$. Thus, annotations can be concatenated.
- Logical postulates are used:
  - If X is an assertion (but not the assumptions) in a legal annotation $\mathcal{A}$, X' is provable from X, and $\mathcal{A}'$ is the result of substituting X' for X in $\mathcal{A}$, then $\mathcal{A}'$ is a legal annotation. Thus, new assertions can be derived from established ones.
  - If X is the assumptions of a legal annotation $\mathcal{A}$, X' is provable from X, and $\mathcal{A}'$ is the result of substituting (X, X') for X in $\mathcal{A}$, then $\mathcal{A}'$ is a legal annotation. Thus, the consequences of the original assumptions can be written down explicitly next to the original assumptions.

A legal annotation of a protocol is much like a sequence of comments about the states of belief of principals in the course of authentication. Step by step, we can follow the evolution from the original assumptions to the conclusions—typically, from the initial beliefs to the final ones.

On time

Note that our logic has not, and does not need, any notion of time to be associated with individual statements. The requirement to deal with time is entirely satisfied by the semantics of the constructs themselves. This is possible because we found it sufficient to reason with stable formulas, that is, formulas that stay true for the whole run of the protocol once they become true. In addition, we represent protocols as sequential algorithms and ignore concurrency issues. As in the published protocols, a partial ordering of algorithmic steps is imposed by functional dependence.

It was a conscious decision to avoid the explicit use of time in the logic presented, and we found it unnecessary for describing the protocols investigated so far. We feel that, though this approach might seem simple-minded, it has greatly increased the ease with which the logic can be manipulated. More ambitious proofs may require the introduction of time, possibly with the use of temporal operators (e.g., [NP], [HV]). However, it is not clear that such proofs would offer greater insight into the workings of the protocols, nor be simple enough to construct without considerable expertise.

It should be noted that some authentication protocols make use of time when choosing nonces, but this does not require time to be made explicit in the logic. This is because the time component can always notionally be replaced by any random number that is known to be shared between the two parties involved, and this can always be accomplished by means of an additional unencrypted message.

The Goal of Authentication, Formalized

Preconditions or assumptions must invariably be introduced at the start of each protocol annotation to describe the initial state of the system. Typically, the assumptions state what keys are initially shared between the principals, which principals have generated fresh nonces, and which principals are trusted in certain ways. Given the assumptions, the verification of a protocol amounts to proving that some formulas hold as postconditions.

There is room for debate as to what should be the goal of authentication protocols that these postconditions describe. We take the following: authentication is complete between A and B if there is a K such that

$$
\begin{aligned}
&A \models A \overset{K}{\leftrightarrow} B\\
&B \models A \overset{K}{\leftrightarrow} B\\
&A \models B \models A \overset{K}{\leftrightarrow} B\\
&B \models A \models A \overset{K}{\leftrightarrow} B
\end{aligned}
$$

The first two of these requirements are essential. The other two are somewhat controversial. In any case, common belief in the goodness of K is never required— A and B need not believe that they both believe that they both believe that ... they both believe that K is good. In fact, some protocols may only attain very weak goals, as for example $A \models B \models X$, for some X, which only reflects that A believes that B has recently sent messages and exists at present.

In addition, these are examples of the sort of questions we would like to be able to answer with the help of formal methods:

Does this protocol get to the goal?

If not, can one proceed from the end of it to the goal?

Does this protocol need more assumptions than another one?

Does this protocol produce more knowledge than needed to meet the goal?

Does this protocol do anything unnecessary that could be left out without weakening the conclusion?

Does this protocol encrypt something that need not be encrypted?

The Otway & Rees Protocol

This protocol was published by Otway and Rees in 1987 ([OR]). We give it below, with A and B as two principals, K_{as} and K_{bs} as their private keys, S as the authentication server; N_a, N_b, and M are nonces, and K_{ab} is the eventual shared secret which will be the conversation key.

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

A passes to B some encrypted material only useful to the server, together with enough information for B to make up a similar encrypted message. B forwards both to the server, who decrypts and checks that the components M, A, B match in the encrypted messages. If so, S generates K_{ab} and embeds it in two encrypted messages, one for each participant, accompanied by the appropriate nonces. Both are sent to B, who forwards the appropriate part to A. Then A and B decrypt, check their nonces, and if satisfied proceed to use K_{ab}.

Now we transform the protocol. The nonce N_c corresponds to M, A, B in the protocol description above.

Message 1 $A \rightarrow B$: $\{N_a, N_c\}_{K_{as}}$

Message 2 $B \rightarrow S$: $\{N_a, N_c\}_{K_{as}}, \{N_b, N_c\}_{K_{bs}}$

Message 3 $S \rightarrow B$: $\{N_a, (A \overset{K_{ab}}{\leftrightarrow} B), (B \mid\!\sim N_c)\}_{K_{as}}, \{N_b, (A \overset{K_{ab}}{\leftrightarrow} B), (A \mid\!\sim N_c)\}_{K_{bs}}$

Message 4 $B \rightarrow A$: $\{N_a, (A \overset{K_{ab}}{\leftrightarrow} B), (B \mid\!\sim N_c)\}_{K_{as}}$

The idealized messages correspond quite closely to the messages described in the published protocol. The main differences can be seen in messages 3 and 4. The concrete protocol description specifies a key and a principal name, which in this sequence have been replaced by the statement that the key can be used to communicate with the principal. This interpretation of the message is possible only because we know how the information in the message is going to be understood. Even more interesting is the statement $A \mid\!\sim N_c$. This does not appear to correspond to anything in the concrete protocol; it represents the fact that the message is sent at all, because if the common nonces had not matched nothing would ever have happened. At this point, we may want to convince ourselves that the idealized protocol accurately represents the actual one and that the guidelines for constructing idealized protocols are not violated.

The protocol analyzed

To analyze this protocol, we first give the assumptions:

$A \models A \overset{K_{as}}{\leftrightarrow} S$

$S \models A \overset{K_{as}}{\leftrightarrow} S$

$B \models B \overset{K_{bs}}{\leftrightarrow} S$

$S \models B \overset{K_{bs}}{\leftrightarrow} S$

$S \models A \overset{K_{ab}}{\leftrightarrow} B$

$A \models (S \Rightarrow A \overset{K}{\leftrightarrow} B)$

$B \models (S \Rightarrow A \overset{K}{\leftrightarrow} B)$

$A \models (S \Rightarrow (B \mid\!\sim X))$

$B \models (S \Rightarrow (A \mid\!\sim X))$

$A \models \sharp(N_a)$

$B \models \sharp(N_b)$
$A \models \sharp(N_c)$

The first four are about shared keys between the clients and the server. The fifth indicates that the server initially knows a key which is to become a shared secret between A and B. The next two assumptions make explicit the trust that A and B have in the server's ability to generate a good encryption key. The next two indicate that each client trusts the server to forward a message from the other client honestly. The final three assumptions show that three nonces have been invented by various principals and are considered to be fresh.

Once we have the assumptions and the idealized version of the protocol, we can proceed to verify it. The rest of the procedure consists merely of applying the postulates of the logic and the annotation rules to the formulas available. It would be excessive to give the detailed deductions that we have checked mechanically, but the steps may be briefly outlined as follows.

A sends his message to B. Now B sees the message, but does not understand it:

$$B \triangleleft \{N_a, N_c\}_{K_{as}}.$$

B is able to generate a message of the same form and to pass it on to S along with A's message. On receiving the message, S can decrypt each encrypted part according to the message meaning postulate, and so deduce that both A and B have encrypted the nonce N_c in their packets:

$$S \models A \mid\!\sim (N_a, N_c), \qquad S \models B \mid\!\sim (N_b, N_c).$$

Note that S cannot tell whether this message is a replay or not, since there is nothing in the message that it knows to be fresh. S emits a message containing two encrypted packets to B. One of the parts is intended for A, and B passes it on.

At this point, both A and B have received a message from the server containing a new encryption key and a nonce. A and B successively apply the postulates on message meaning, nonce verification, and jurisdiction, and emerge with the following final beliefs:

$A \models A \overset{K_{ab}}{\leftrightarrow} B$
$A \models B \models N_c$
$B \models A \overset{K_{ab}}{\leftrightarrow} B$
$B \models A \mid\!\sim N_c$

This is far from being complete authentication as specified above. The authentication could be completed by handshaking between A and B; the weakness is that the key K_{ab} has never been used, and so neither principal can know whether the key is known to the other. A is in a slightly better position than B, in that A has been told that B emitted a packet containing a nonce that A believes to be fresh. This allows A to infer that B has sent a packet recently—B exists. B has been told by the server that A has used a nonce, but B has no idea whether this is a replay of an old message or not.

In addition there are various forms of redundancy in the protocol. Two nonces are put up by A; however the verification using N_a could just as well have been done using N_c, and N_a is redundant. Furthermore there is redundant encryption. N_a is redundant in the first message, and N_b need not be encrypted in the second. As these possibilities are explored, we rapidly move towards a protocol of different structure which would have different concrete realizations.

The Needham & Schroeder Protocol

This protocol was published by Needham and Schroeder in 1978 ([NS1]). The cast of players is the same as in the Otway & Rees protocol.

Message 1	$A \rightarrow S$:	A, B, N_a
Message 2	$S \rightarrow A$:	$\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_b}\}_{K_a}$
Message 3	$A \rightarrow B$:	$\{K_{ab}, A\}_{K_b}$
Message 4	$B \rightarrow A$:	$\{N_b\}_{K_{ab}}$
Message 5	$A \rightarrow B$:	$\{N_b - 1\}_{K_{ab}}$

Here only A makes contact with the server, who provides A with the conversation key, K_{ab}, and a certificate encrypted with B's key conveying the conversation key and A's identity to B. Then B decrypts this certificate and carries out a nonce handshake with A to be assured that A is present currently, since the certificate might have been a replay. The use of $N_b - 1$ in the last message is conventional. Almost any function of N_b would do, as long as B can distinguish his message from A's—thus, subtraction is used to indicate that the message is "signed."

The idealized protocol is as follows:

Message 1	$A \rightarrow S$:	N_a
Message 2	$S \rightarrow A$:	$\{N_a, (A \overset{K_{ab}}{\leftrightarrow} B), \sharp(A \overset{K_{ab}}{\leftrightarrow} B), \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_b}\}_{K_a}$
Message 3	$A \rightarrow B$:	$\{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_b}$
Message 4	$B \rightarrow A$:	$\{N_b, (A \overset{K_{ab}}{\leftrightarrow} B)\}_{K_{ab}}$ *signed* B
Message 5	$A \rightarrow B$:	$\{N_b, (A \overset{K_{ab}}{\leftrightarrow} B)\}_{K_{ab}}$ *signed* A

As in the Otway & Rees protocol, the addresses in the first message have been combined with the nonce for simplicity. The last two messages of the idealized protocol are written in full, with their signatures. This is merely to distinguish the two messages, which might otherwise be confused. In this case, the concrete realization of the signature is, of course, the subtraction in the final message.

The additional statements about the key K_{ab} in the second and in the last two messages are present to assure A that the key can be used as a nonce and to assure each principal that the other believes the key is good. These statements can be included because neither message would have been sent if the statements were not believed.

The protocol analyzed

To start, we give some assumptions:

$$A \mid\equiv A \overset{K_{as}}{\leftrightarrow} S$$
$$S \mid\equiv A \overset{K_{as}}{\leftrightarrow} S$$
$$B \mid\equiv B \overset{K_{bs}}{\leftrightarrow} S$$
$$S \mid\equiv B \overset{K_{bs}}{\leftrightarrow} S$$
$$S \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$$

$$A \mid\equiv (S \Rightarrow A \overset{K}{\leftrightarrow} B)$$
$$B \mid\equiv (S \Rightarrow A \overset{K}{\leftrightarrow} B)$$
$$A \mid\equiv (S \Rightarrow \sharp(A \overset{K}{\leftrightarrow} B))$$

$$A \mid\equiv \sharp(N_a)$$
$$B \mid\equiv \sharp(N_b)$$
$$S \mid\equiv \sharp(A \overset{K_{ab}}{\leftrightarrow} B)$$

Most of the assumptions are routine. Three indicate exactly what the clients trust the server to do. As before, S is trusted to make new keys for A and B, but here A also trusts S to generate a key which has the properties of a nonce. In fact, one can argue that a good encryption key is very likely to make a good nonce in any case. However, the need for this assumption has highlighted the need for this feature in the protocol.

The next assumption is unusual:

$$B \mid\equiv \sharp(A \overset{K}{\leftrightarrow} B).$$

The protocol has been criticized for using this assumption ([DS]), and the authors did not realize they were making it. The proof outlined below shows how this added assumption is needed to attain authentication.

Again the detail in the verification is suppressed. First, A sends a cleartext message containing a nonce. This can be seen by the server, who repeats the nonce in the reply. The reply from S also contains the new key to be used between A and B. Then A sees the entire message,

$$A \triangleleft \{N_a,\ (A \overset{K_{ab}}{\leftrightarrow} B),\ \sharp(A \overset{K_{ab}}{\leftrightarrow} B),\ \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}\}_{K_{as}},$$

which it decrypts using the message meaning postulate. Since A knows N_a to be fresh, we can also apply the nonce verification postulate, leading to:

$$A \mid\equiv S \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B, \qquad A \mid\equiv S \mid\equiv \sharp(A \overset{K_{ab}}{\leftrightarrow} B).$$

The jurisdiction postulate allows A to infer:

$$A \models A \overset{K_{ab}}{\leftrightarrow} B, \qquad A \models \sharp(A \overset{K_{ab}}{\leftrightarrow} B).$$

Also, A has seen the part of the message encrypted under B's private key,

$$A \triangleleft \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{b*}}.$$

This allows A to send this as a message to B. At this point, B can use the message meaning postulate to decrypt the message,

$$B \models S \mid\sim A \overset{K_{ab}}{\leftrightarrow} B.$$

Unlike A, however, B is unable to proceed without resorting to the dubious assumption set out above. B knows of nothing in the message which is fresh, so it cannot tell when this message was generated. B simply assumes that the message from the server is fresh.

If we allow B to make the necessary assumption, the rest of the protocol proceeds without problem. B can immediately obtain the key,

$$B \models A \overset{K_{ab}}{\leftrightarrow} B,$$

via the postulates of nonce verification and jurisdiction.

The last two messages cause A and B to become convinced that the other exists (that is, he has sent messages recently) and is in possession of the key. B first encrypts its nonce and sends it to A, who can deduce that B believes in the key,

$$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B,$$

because he has been guaranteed the freshness of the key by S. Then A replies similarly, and B can deduce that A also believes in the key,

$$B \models A \models A \overset{K_{ab}}{\leftrightarrow} B.$$

Note that the freshness of the nonce N_b is sufficient for B to deduce this. It is not necessary to reuse the dubious assumption.

This results in the following beliefs:

$A \models A \overset{K_{ab}}{\leftrightarrow} B$
$B \models A \overset{K_{ab}}{\leftrightarrow} B$
$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$
$B \models A \models A \overset{K_{ab}}{\leftrightarrow} B$

In fact, we even obtain $B \models A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$.

This is a stronger outcome than in the Otway & Rees protocol, but it is reached at the cost of the extra assumption that B accepts the key as new. Denning and Sacco pointed out that compromise of a conversation key can have very bad results: an intruder has unlimited time to find an old session key and to reuse it as though it were fresh. It is comforting that the logical analysis makes explicit the assumption.

Clearly, the problem is that B has no interaction with S that starts with B's initiative. It is possible to rectify this by starting with B rather than A, and this was done by Needham and Schroeder in 1987 ([NS2]). The note by Needham and Schroeder was published adjacent to the paper by Otway and Rees. Perhaps for the lack of a calculus to describe these protocols, none of the people involved realized that the proposals were essentially the same, though Needham and Schroeder went on to do the complete job, rather than leaving the final stages to be combined with the first transmissions of data as in the design by Otway and Rees.

It may also be noticed that in message 4 a nonce is being sent encrypted when this is in general not necessary. However, in this instance, if the nonce were sent unencrypted it would be necessary to send something else encrypted, because subsequent deductions rely on an inference about key values made in the message meaning postulate, which only applies to encrypted messages.

Semantics

Hopefully, the formulas liberally sprinkled throughout this paper are intuitively clear. In this section we take a more formal approach than previously and discuss a formal semantics of these formulas.

Beliefs

We describe an "operational" semantics. According to this semantics, principals develop beliefs by computation. In order to obtain new beliefs, principals are supposed to examine their current beliefs and apply a few computationally tractable inference rules. These rules represent the idealized workings of principals in actual authentication protocols. Thus, the statement that an authentication protocol gives rise to certain beliefs is a strong one: it means that the principals develop these beliefs even with realistic computational resources. In contrast, a restrictive, operational notion of belief would certainly be harmful in the study of security properties, where we would want to guarantee that intruders learn no secrets even with unknown methods or algorithms.

The *local state* of a principal P is two sets of formulas $\mathcal{S}_P$ and $\mathcal{B}_P$. Intuitively, $\mathcal{S}_P$ is the set of messages that the principal sees and $\mathcal{B}_P$ is the set of beliefs of the principal. The sets $\mathcal{S}_P$ and $\mathcal{B}_P$ enjoy some closure properties that correspond directly to the inference rules of the logic. For instance,

$$\text{if } (P \overset{K}{\leftrightarrow} Q) \in \mathcal{B}_P \text{ and } \{X\}_K \in \mathcal{S}_P \text{ then } X \in \mathcal{S}_P.$$

We imagine the closure properties are enforced by algorithms to derive and add new messages and beliefs.

A *global state* is a tuple containing the local states of all principals; in all the cases we consider, it is a triple with the local states of A, B, and S. If s is a global state then s_P is the local state of P in s and $\mathcal{B}_P(s)$ and $\mathcal{S}_P(s)$ are the corresponding sets of beliefs and messages. The *satisfaction* relation between global states and formulas has a trivial definition: $P \models X$ holds in state s if $X \in \mathcal{B}_P(s)$, and $P \lhd X$ holds if $X \in \mathcal{S}_P(s)$. A set (or conjunction) of formulas holds in a given state if each member holds.

A *run* is a finite sequence of states $s_0, \ldots, s_n$ where $\mathcal{B}_P(s_i) \subseteq \mathcal{B}_P(s_{i+1})$ and $\mathcal{S}_P(s_i) \subseteq \mathcal{S}_P(s_{i+1})$ for all $i \leq (n-1)$ and for each principal P. In other words, the sets of messages seen and the sets of beliefs can only increase. A run is a run of a particular protocol if all of the messages the protocol prescribes are communicated—other messages may be initially present or may come from the environment. More precisely, a protocol is a finite sequence of n "send" statements of the form $(P_1 \rightarrow Q_1 : X_1), \ldots, (P_n \rightarrow Q_n : X_n)$; a run of the protocol is a run of length $n+1$ where $X_i \in \mathcal{S}_{Q_i}(s_i)$ for all $i \leq n$.

An annotation for the protocol holds in a run of the protocol if all of the formulas in the annotation hold in the corresponding states. More precisely, an annotation consists of $n+1$ sets of formulas of the forms $P \models X$ and $P \lhd X$ inserted before and after statements; it holds in a run of the protocol if the i-th set holds in the i-th state of the run for all $i \leq n$. An annotation is valid if it holds in all runs of the protocol where the first set of the annotation—the assumptions—holds.

Immediately, the annotation rules described earlier are sound: all legal annotations are valid. The rules are also complete: all valid annotations are legal. To see this, given a protocol $(P_1 \rightarrow Q_1 : X_1), \ldots, (P_n \rightarrow Q_n : X_n)$ consider a run $s_0, \ldots, s_n$ where only the messages $X_1, \ldots, X_n$ are communicated. All valid annotations must hold in this run. Furthermore, we can show that any annotation that holds in this run can be derived.

Truth and true beliefs

While the semantics gives a meaning to the operators $\models$ and $\lhd$, the remaining operators are still largely a mystery. For instance, the semantics does not determine whether $A \mid\!\sim N_a$ is true or false in a given state. This is a deficiency if we are interested in judging the truth of beliefs. In order to give a meaning to the remaining operators, the notion of state needs to be richer than the one we have used so far, as follows:

- Each state associates with each principal P a set $\mathcal{O}_P$ of formulas that it once said. This set has two closure properties: if $(\{X\}_K \textit{ signed } P) \in \mathcal{O}_P$ then $X \in \mathcal{O}_P$; if $(X, Y) \in \mathcal{O}_P$ then $X \in \mathcal{O}_P$. We require that if $P_i \rightarrow Q_i : X_i$ is the i-th action of a protocol then the set of formulas once said increases only for P_i in the i-th state of all runs of this protocol. More precisely, if $R \neq P_i$ then $\mathcal{O}_R(s_i) = \mathcal{O}_R(s_{i-1})$ and $\mathcal{O}_{P_i}(s_i)$ is the closure (by the rules above) of $\mathcal{O}_{P_i}(s_{i-1}) \cup \{X_i\}$. In addition, each principal must believe all the formulas it has said recently, in the sense that if $X \in \mathcal{O}_P(s)$ because of a message in the protocol then $X \in \mathcal{B}_P(s)$.
  The formula $P \mid\!\sim X$ holds in state s if $X \in \mathcal{O}_P(s)$.
- In each run each principal P has jurisdiction over a set of formulas $\mathcal{J}_P$. We require that if $X \in \mathcal{J}_P$ and $P \models X$ holds then X holds as well.
  The states in the run satisfy $P \Rightarrow X$ if $X \in \mathcal{J}_P$.

- Each run assigns a set of good keys $\mathcal{K}_{\{P,Q\}}$ to each pair of principals P and Q. We require that good keys are only used by the appropriate principals, that is, if $R \triangleleft (\{X\}_K \textit{ signed } R')$ and $K \in \mathcal{K}_{\{P,Q\}}$ then either $R' = P$ and $(\{X\}_K \textit{ signed } P) \in \mathcal{O}_P$ or $R' = Q$ and $(\{X\}_K \textit{ signed } Q) \in \mathcal{O}_Q$.

  The states in the run satisfy $P \overset{K}{\leftrightarrow} Q$ if $K \in \mathcal{K}_{\{P,Q\}}$.
- Since we are not concerned with expressions of the form $P \models \{X\}_K$, we do not assign a truth value to expressions of the form $\{X\}_K$—one could be given, but only somewhat artificially.
- Each run determines a set of fresh formulas $\mathcal{F}$. This set has a closure property: if $X \in \mathcal{F}$ and X is a subformula of Y then $Y \in \mathcal{F}$. If $X \in \mathcal{F}$ and X was once said (that is, $X \in \mathcal{O}_P(s_i)$ for some P and i) then X should have been said recently (that is, $X \notin \mathcal{O}_P(s_0)$ for all P).

  The states in the run satisfy $\#(X)$ if $X \in \mathcal{F}$.

Clearly, some beliefs are false. This seems essential to a satisfactory semantics. Questions of trust and delegation, central to our study, would become meaningless if all beliefs had to be true. Moreover, we can consider many interesting runs—for instance, those where an intruder has broken the cryptosystem—because we leave open the possibility of incorrect beliefs.

Let us define *knowledge* as "truth in all possible worlds" (e.g., [HM]). More precisely, P knows X in state s if and only if X holds in all states s' where the local state of P is the same as in s, that is, $s'_P = s_P$. In general, the notions of knowledge and belief are incomparable. For instance, some erroneous initial beliefs are certainly not knowledge, while each principal knows all tautologies, but does not necessarily believe them.

Most beliefs happen to be true in practice, but the semantics does not account for this coincidence. To guarantee that all beliefs are true we would need to guarantee that all initial beliefs are true. In this case, belief is a rudimentary resource-bounded approximation to knowledge.

Conclusions

Previous works on authentication had suggested that notions such as knowledge and belief would be essential to understand authentication protocols. However, a precise analysis had not been undertaken.

Recent literature has emphasized the importance of reasoning about knowledge and action to understanding distributed computation (e.g., [HM]). In fact, there have been some attempts to analyze cryptographic protocols ([DLM], [MW]). Such endeavors could be useful in providing a general theoretical basis for our work. On the other hand, practical applications of epistemic logics to distributed computing problems are still few (e.g., [HZ]).

The examples in this study show how an extremely simple logic can capture subtle differences between different protocols. The logic lacks all features that would make it difficult to use yet it does what is needed—it enables us to exhibit step by step how beliefs are built up to the point of mutual authentication.

Acknowledgements

The work was undertaken as the result of a suggestion by Butler Lampson. Andrew Birrell and Michael Schroeder made valuable contributions. Luca Cardelli and Dorothy Denning suggested improvements in the presentation of this work. Kathleen Sedehi typeset the paper.

References

[DLM] R.A. DeMillo, N.A. Lynch, and M.J. Merritt. Cryptographic Protocols. *Proceedings of the Fourteenth ACM Symposium on the Theory of Computing*, 1982, pp. 383–400.

[DS] D.E. Denning and G.M. Sacco. Timestamps in Key Distribution Protocols. *CACM* Vol. 24, No. 8, August 1981, pp. 533–536.

[H] C.A.R. Hoare. An Axiomatic Basis for Computer Programming. *CACM* Vol. 12, No. 10, October 1969, pp. 576–580.

[HM] J.Y. Halpern and Y.O. Moses. Knowledge and Common Knowledge in a Distributed Environment. *Proceedings of the Third ACM Conference on the Principles of Distributed Computing*, 1984, pp. 480–490.

[HV] 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*, 1986, pp. 304–415.

[HZ] J.Y. Halpern and L. Zuck. A Little Knowledge Goes a Long Way: Simple knowledge-based derivations and correctness proofs for a family of protocols. Yale University technical report DCS/TR 517, February 1987.

[MCF] J.K. Millen, S.C. Clark, and S.B. Freedman. The Interrogator: Protocol Security Analysis. *IEEE Transactions on Software Engineering* Vol. SE-13, No. 2, February 1987, pp. 274–288.

[MW] M.J. Merritt and P.L. Wolper. States of Knowledge in Cryptographic Protocols. Draft.

[NP] V. Nguyen and K.J. Perry. Do We Really Know What Knowledge Is? Draft.

[NS1] R.M. Needham and M.D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *CACM* Vol. 21, No. 12, December 1978, pp. 993–999.

[NS2] R.M.Needham and M.D. Schroeder. Authentication Revisited. *Operating Systems Review* Vol. 21, No. 1, January 1987, p. 7.

[OR] D. Otway and O. Rees. Efficient and Timely Mutual Authentication. *Operating Systems Review* Vol. 21, No. 1, January 1987, pp. 8–10.